FineType: Fine-grained Tapping Gesture Recognition for Text Entry

Chentao Li
lict23@mails.tsinghua.edu.cn
Department of Automation, BNRist, Tsinghua University
Beijing, China

Ziheng Xi
xizh21@mails.tsinghua.edu.cn
Department of Automation, BNRist, Tsinghua University
Beijing, China

Jianjiang Feng*
jfeng@tsinghua.edu.cn
Department of Automation, BNRist, Tsinghua University
Beijing, China

Jie Zhou
jzhou@tsinghua.edu.cn
Department of Automation, BNRist, Tsinghua University
Beijing, China

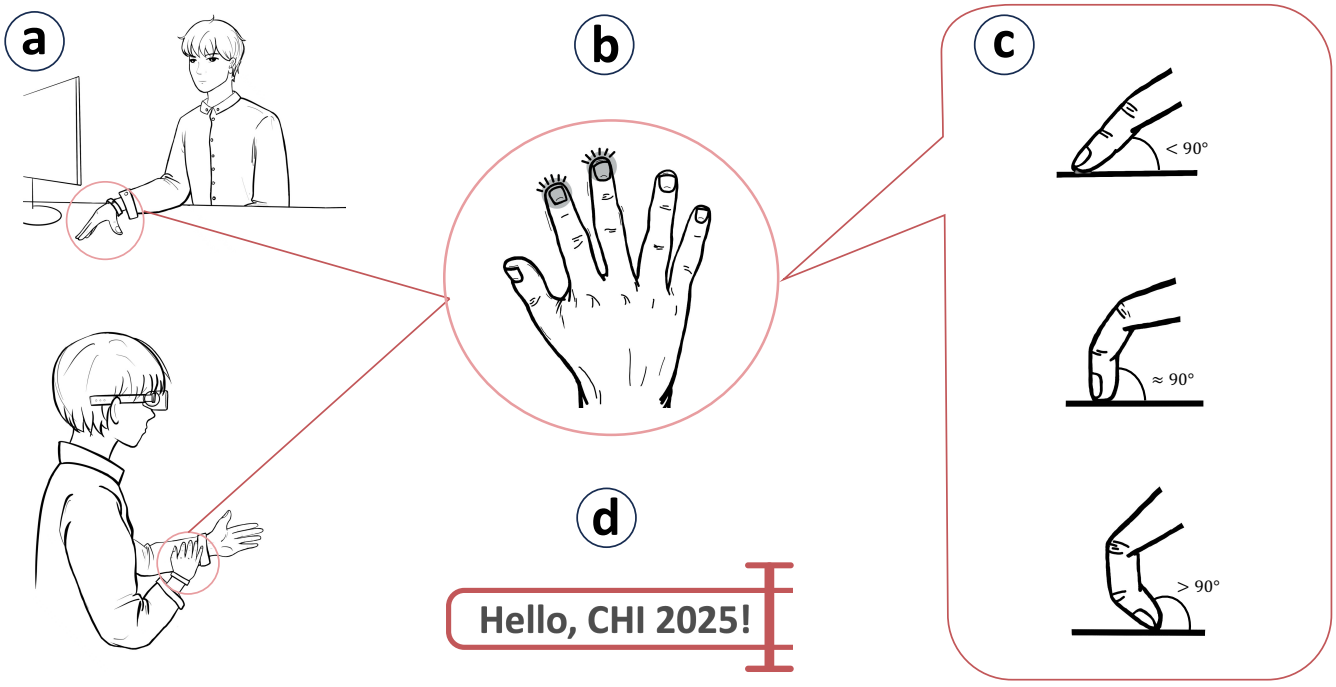


Figure 1: Diagram of the core concept of *FineType*. (a) A user wears the wristband device and can perform eyes-free text input while using AR glasses or working on a regular computer by tapping their fingers on a flat surface (desk, arm). The tapping gestures are primarily composed of (b) different finger combinations and (c) three types of tapping finger postures. *FineType* allows for versatile text input, mapping tapping gestures to letters, numbers, and symbols, as shown in (d).

ABSTRACT

With the rise of mixed reality (MR) and augmented reality (AR) applications, efficient text input in AR/MR environments remains challenging. We propose *FineType*, a text entry system using tapping gestures with finger combinations and postures on any flat surface. Using a wristband with an IMU and an infrared camera, we detect tapping events and employ a multi-task convolutional neural network to predict these gestures, enabling nearly full keyboard mapping (including letters, symbols, numbers, etc.) with one hand. We collected gestures from participants (N=28) with 10 finger combinations and 3 finger postures for training. Cross-user validation showed accuracies of 98.26% for combinations, 95.53% for postures, and 94.19% for all categories. For 8 newly defined finger combinations and their postures, classification accuracies were 91.27% and 93.86%. Using user-adaptive few-shot learning, we improved the

*Corresponding author

CHI '25, April 26-May 1, 2025, Yokohama, Japan

ACM ISBN 979-8-4007-1394-1/25/04
https://doi.org/10.1145/3706598.3714278

finger combination accuracy to 97.05%. The results demonstrate our potential to map tapping gestures composed of all finger combinations and three postures. Our user study (N=10) demonstrated an average typing speed of 35.1 WPM with a character error rate of 5.1% after two hours of practice.

CCS CONCEPTS

• **Human-centered computing → Keyboards; Text input**.

KEYWORDS

text entry; deep nerual network; finger posture; finger combination; interaction techniques

ACM Reference Format:

Chentao Li, Ziheng Xi, Jianjiang Feng, and Jie Zhou. 2025. FineType: Fine-grained Tapping Gesture Recognition for Text Entry. In *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26-May 1, 2025, Yokohama, Japan.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3706598.3714278

1 INTRODUCTION

With the rise of mixed and augmented reality (MR, AR) head-mounted devices (HMDs), such as Meta Quest, Apple Vision Pro, and Microsoft HoloLens, the realization of spatial computing is becoming possible, paving the way for modern computing and more advanced human-computer interaction [10, 21]. Text entry, as a primary method of interacting with computing devices, is a basic need when interacting with HMDs. While physical keyboards have been the standard in the PC era due to their high typing efficiency, their lack of portability makes them impractical for mobile computing. In contrast, touchscreen keyboards on mobile devices like smartphones, tablets, and smartwatches provide convenient typing but come with trade-offs, such as reduced accuracy, smaller size, and limited comfort [16, 28, 69, 72]. Touchscreen text entry method relies on visual feedback, requiring frequent gaze shifts between the text area and keyboard while still occupying screen space despite its compact design [59, 60, 84]. As we move into the era of spatial computing and become increasingly detached from PCs and mobile devices, the need for text entry will arise in various scenarios. However, there is currently no dominant solution for text input in spatial computing environments [79, 82]. Existing options either have slow input speeds, which reduce user efficiency in activities like messaging and note-taking [17]; require mid-air typing [12, 30, 50] without tactile feedback, leading to fatigue and inefficiency [11, 27, 35]; or depend on visual feedback with complex multi-sensor setups and computing resources to track hand tapping positions [13], making eyes-free operation impossible. Although voice input is a viable option, it can pose privacy risks and attract unwanted attention in quiet environments [14, 30, 39].

Revisiting the physical keyboard, it has always been the preferred choice for prolonged use and heavy text input. One of its key advantages is its well-organized layout, which distributes a large number of keys across multiple rows. This allows users to easily and comfortably reach keys with minimal finger movement and almost no wrist movement, enabling fast typing [9, 15]. Another benefit is its tactile feedback, combined with a flat surface for key presses, which supports prolonged use [27, 62]. We noticed that when typing on a physical keyboard, users mainly rely on different fingers and specific finger movements to press the keys. If we abstract these fingers and movements into discrete gestures, then the keyboard can be seen as defining each key through a combination of ten fingers and specific finger postures. This insight inspires us to explore defining different keys based on the gesture of the tapping finger.

In this paper, we present *FineType*, a one-handed typing system that enables users to type on arbitrary flat surfaces, mapping nearly all keyboard keys, including letters, symbols, and numbers. To achieve this, we were inspired by the multi-row layout of physical keyboards and defined three postures for finger tapping on a surface (see Fig. 1): middle fingertip touch, front fingertip touch, and fingernail touch, where the corresponding angles between the first finger joint and the surface are $< 90°$, $\approx 90°$, and $> 90°$, respectively. Meanwhile, *FineType* used individual fingers and combinations of different fingers to represent the letters on each row of the keyboard. We primarily focus on one-handed input, as it provides superior portability and flexibility in spatial computing environments. This approach allows users to keep one hand free, making it well-suited for a broader range of interaction scenarios. Moreover, related work has shown that using fingers [54, 83] or fingernail [32] to touch surfaces can effectively increase the efficiency of touch interaction. Further studies, which use finger orientation [25, 41, 51, 70, 75] to modify touch input, demonstrates the rationale for expanding the tapping gesture space using finger posture. By combining three finger postures with various finger combinations to create multiple finger tapping gestures, we can nearly achieve a complete mapping of all keys on a full-sized keyboard. Utilizing a wristband equipped with an IMU and an infrared camera below the wrist, *FineType* detects finger taps and captured the infrared images of finger tapping gestures in real-time. Subsequently, *FineType* employs a detection model based on convolutional neural networks for recognizing finger postures and combinations. In contrast to TapXR [34, 58] device, which consider only different finger combinations, *FineType* further takes finger postures into account, analogous to a keyboard, expanding the input space threefold and greatly enhancing the variety of command set. Furthermore, while TapXR's technical details and raw sensor data are closed off due to its commercial use, we present the technical details of the hardware platform and recognition algorithm.

Recent studies on text input through finger tapping on flat surfaces have gained significant attention. TapID [52] and TapType [62] use wrist-mounted IMUs to detect finger taps, while TypeAnywhere [82] utilizes TapStrap [33]. Although these methods adopt the QWERTY layout for easier user transition, they map ten fingers to 26 letters, lacking a one-to-one mapping. Consequently, they require complex language models to decode typing sequences, which struggle with out-of-vocabulary (OOV) words or necessitate additional gestures for selection [62], ultimately reducing text input speed. In contrast, *FineType* encodes input based solely on the index of the tapping finger and its posture, without relying on spatial position. This enables a single hand to map up to $(2^5 - 1) \times 3 = 93$ symbols, allowing one-handed text input with nearly as many keys as a full-sized keyboard, regardless of device type or touch area.

We designed a multi-task convolutional neural network to detect fine-grained tapping gestures. The model consists of three branches:

predicting finger postures, detecting multi-label finger-tapping combinations, and regressing fingertip heatmaps. Cross-user validation (N=28) demonstrated an accuracy of 98.26%, 95.53%, and 94.19% for 10 finger combinations, 3 finger postures, and all 30 fine-grained gestures, respectively. Furthermore, we proposed a user few-shot learning method to improve the accuracy of gestures not present in the training set. Additionally, ablation studies further confirmed the advantages and rationality of our model design.

In an online text entry study (N=10), after 2 hours of practice, participants using *Finetype* achieved an average of 35.1 WPM and 5.1% character error rate on transcription tasks containing only English letters, which is 93% of the one-handed touchscreen input speed. For a more comprehensive phrase set containing letters, numbers, and symbols, they achieved 15.0 WPM and 7.5% character error rate. *FineType* outperformed TapXR in text entry speed and demonstrated lower error rates on complex symbol sets. Subjective user feedback and quantitative analysis also confirmed the comfort and typing performance of *FineType*.

In summary, our paper makes the following core contribution:

- We present a one-handed text input system using wearable wrist sensors to detect fine-grained tapping gestures on flat surfaces, combining three distinct finger-tapping postures with finger combinations. This approach enables one-handed mapping of letters, numbers, and symbols, providing functionality comparable to a full-sized keyboard.
- We propose a convolutional neural network architecture capable of recognizing up to 93 fine-grained tapping gestures by decoding finger combinations and tapping postures. Offline cross-user validation and ablation studies confirmed the effectiveness of our network design.
- We conducted a text entry user study (N=10) to evaluate text input performance on the MacKenzie and Soukoreff's phrase set [47] and a more comprehensive character set.

2 RELATED WORK

This work mainly focuses on hand-based text input, so we reviewed hand-based text input techniques proposed or potentially usable for XR systems. Table 1 summarizes some representative methods and their performance.

2.1 Hand-worn Devices

Another forms of wearable input are hand-worn devices equipped with various sensors. KITTY [40] and DigiTouch [73] utilize electronic contacts or partially conductive fabric strips on gloves to detect finger touch events, achieving typing speeds of 1.8–5 WPM and 16 WPM, respectively. PrinType [45] uses different finger regions as a virtual keyboard in HMD VR and achieves 34.22 WPM with a Bayesian decoder, but its use of fingerprint sensors may be affected by surface conditions like finger moisture. PrinType also requires a time-consuming fingerprint registration process and can only be used by the registered user. QwertyRing [23] utilizes an IMU to detect typing movements with the index finger on a flat surface, estimating characters based on finger direction. TapStrap [33] is a commercial typing accessory that attaches an inertial sensor ring to each finger to detect character input on flat surfaces. Typeanywhere [82], based on TapStrap, has designed a language model that decodes typing sequences based on finger tapping sequences. Another similar device, Telemetring [64] detects finger tapping events using a telemetry sensing method and uses chords for text input. It achieved a recognition rate of 89.7% for 31 finger combinations, but it only evaluated one person and did not report typing speed. Lian et al. [42] proposed an arc-shaped keyboard layout with large angular differences between adjacent keys, using motion-sensing rings on each hand to detect inputs. This design maps up to 15 keys per hand but requires precise positioning and has not been tested in typing experiments. Furthermore, using all ten fingers with decoder methods [62, 82], although mimicking traditional keyboard typing habits, does not achieve precise mapping to specific letters, making it difficult for users to type non-word characters (like passwords and phone numbers). *FineType* addresses this issue by introducing three finger postures combined with different finger combinations, providing three precise encodings for each finger combination. This maximizes distinguishable finger tapping gestures and significantly expands the mappable character set.

2.2 Wrist-worn Devices

Wearing a wristband device offers a portable text input method that does not hinder hand interactions. PalmType [71] uses the palm as a virtual display for smart glasses, achieving a typing speed of 7.7 WPM. DigiTap [55] employs a wristband with a camera and accelerometer to detect thumb-to-finger tapping, reaching 10 WPM. Similar to DigiTap's hardware, our setup replaces the LED flash with an infrared camera, enabling input without visible light, making it better suited for privacy-sensitive scenarios. However, such methods often result in low typing speeds, limiting their practicality. WrisText [19] uses wrist movements, detected by infrared sensors in a wristband, to mimic joystick controls, achieving 9.9 WPM. ViFin [6] utilizes wrist-worn IMUs to detect vibrations from mid-air finger writing, while TapID [52] detects touch events on passive surfaces using an accelerometer, aiding VR input. TapType [62] enhances TapID's accuracy by eliminating VR dependencies with a Bayesian classifier. TapXR [58] uses a visual sensor to predict finger combinations but does not fully leverage visual analysis for advanced hand feature detection. Niikura et al. [53] used a wrist-mounted camera and contact microphone to distinguish light and heavy taps, achieving ~77% accuracy for 36 gestures under desk conditions, but requiring ambient noise calibration and lacking a text entry study. Chen et al. [7] used IMUs to detect finger taps on the back of the hand, mapping a numeric keypad and symbols with 93.89% accuracy across 12 positions, but without reporting typing speed. This approach requires precise tapping force and accuracy, limiting its ease of use. Additionally, ShadowTouch [43] uses light sources beneath the wrist to illuminate finger shadows, enabling the inference of the touch state of each finger.

2.3 Mid-air Typing Based on Head-worn Camera

Another emerging approach in text input technology is HMD-based mid-air typing. Yu et al. [78] utilized an HMD's directional tracking to enable gesture typing based on head orientation, achieving an average typing speed of 24.7 WPM. The VISAR keyboard, designed for HMD-AR scenarios, employed a statistical decoder to reach 17.75

Table 1: Hand-based text input methods and sensing technologies proposed or potentially usable in XR scenarios, along with their words per minute (WPM) and error rate (%). ('*' in the performance indicates that the method relies on a decoder for input. '-' indicates that the method does not report the performance.) The table also includes the number of supported keys or gestures, whether eyes-free typing is possible, and whether out-of-vocabulary (OOV) words are supported.

Category	Study	Year	Sensor	Number of Keys / Gestures	Eyes-free	Support OOV words	Speed(WPM)	Error(%)
Head-worn	Yu et al. [78]	2017	VR HMD	26	✗	✗	24.73*	0.57*/2.46*/1.13*
	VISAR [12]	2018	AR HMD	27	✗	✗	17.8*	0.6*
	Adhikary and Vertanen [2]	2021	VR HMD + LeapMotion	28	✗	✔	16/15	1
	ThumbAir [18]	2023	VR HMD	8	✗	✗	13.73*	1.2*
	Fu et al. [17]	2024	AR HMD	27	✗	✔	40	9
Hand-worn	DigiTouch [73]	2017	Touch gloves	28	✔	✔	16	0.85
	QwertyRing [23]	2020	IMU	26	✔	✗	13.74-20.59*	1.17*
	TelemetRing [64]	2020	Telemetry	31	✔	✔	-	10.3
	PrinType [45]	2022	Fingerprint sensor	up to 65	✔	✔	29.56*/32.38*/34.22*	8.78
	TypeAnywhere [82]	2022	TapStrap (10 IMUs)	14	✔	✗	70.6*	1.50
Wrist-worn	DigiTap [55]	2014	IMU+Camera	12	✔	✗	10	6.3
	PalmType [71]	2015	IR sensor	28	✔	✔	10.01	1.58
	WristText [19]	2018	IR sensor	6	✔	✗	9.9*	5.92*
	ViFin [6]	2021	IMU	36	✔	✔	16	13.5
	TapType [62]	2022	IMU	12	✔	✔	19*	0.6*
	Chen et al. [7]	2023	IMU	12	✔	✗	-	6.11/6.01/5.60
	Ours	-	IR Camera + IMU	up to 93	✔	✔	35.1	5.1

WPM [12]. MRTouch [76] leveraged infrared and depth cameras in AR to track hand postures, enabling precise clicks with reduced positional errors. Fu et al. [17] predicted key presses from user-perspective video streams captured by AR headsets, achieving 40 WPM. ThumbAir [18] demonstrated a mid-air typing method using thumb movements, averaging 13.7 WPM with a 1.2% error rate, while STAR [37] introduced a virtual QWERTY keyboard overlaid on the skin, achieving 21.9 WPM. However, these methods often rely on AR/VR cameras requiring full hand visibility, making them unsuitable in cases of occlusion. While mid-air text input methods offer promise, prolonged use can lead to discomfort [27, 35]. Recent advancements using head-mounted displays include StegoType [56], achieving 42.4 WPM with unmarked egocentric hand tracking, and TouchInsight [63], reaching 37 WPM with uncertainty-aware touch detection on physical surfaces.

2.4 Text Entry Based on External Camera

Some methods leverage body-worn or external cameras to recognize hand movements or gestures for text input. Markussen [49, 50] introduced the Vulture mid-air gesture keyboard, which uses a marker-based system to track hands, achieving 21 WPM. TypeNet [48] relies on an RGB camera placed on a flat surface to train neural networks for capturing typing video frames and letters, achieving 93.5% character-level accuracy. However, it requires precise camera placement, struggles in low-light conditions, and incurs high power consumption due to continuous operation. Richardson et al. [57] employed OptiTrack cameras to capture hand movements and trained temporal convolutional networks to decode these into text. ATK [77] and studies by Adhikary and Vertanen [2] utilized Leap Motion sensors to detect 3D hand postures and enable mid-air typing on QWERTY keyboards. However, the lack of tactile feedback in these systems often leads to fatigue during prolonged use [35]. PressureVision++ [20] estimated fingertip pressure intensity with RGB cameras, offering tactile feedback through physical surface touch and achieving 25.8 WPM, which is 78% faster than gesture-based mid-air keyboards. These external camera-based methods often face challenges with obstruction, precise sensor placement, and limited adaptability to different environments. In contrast, *FineType* uses an infrared camera beneath the wrist to capture finger tapping and contact postures, tripling the character set while adapting to various lighting conditions. Combined with an IMU sensor, the camera activates only when fingers move, significantly reducing power consumption.

3 *FINETYPE* OVERVIEW

We briefly introduce the set of finger gestures used to train our model, the design of our hardware prototype, and a high-level overview of the system.

3.1 Training Gesture Set

We have defined three finger postures and ten finger combinations, and included a category for non-tap images with the surface. Fig. 2 displays the gesture set used for training, along with real images captured by our wristband device under each category. We trained the model using only ten tapping finger combinations, but we expect that training with just these combinations will enable the model to distinguish all possible tapping combinations (i.e. $2^5 - 1 = 31$). This design leads to two questions: **Q1**: Why were these three postures selected? **Q2**: Why were these finger combinations chosen?

To answer **Q1**, physical keyboards have a multi-row layout which has proven to enable prolonged use for efficient text input [27], indicating that users can tolerate fingers bending to different degrees with low fatigue. Additionally, users are highly sensitive to the perception of their hands and can identify them without visual attention [45]. We defined three finger contact postures with the surface: mid-fingertip contact, front-fingertip contact, and nail contact. These three gestures are easy for users to distinguish and operate. Each finger posture we defined is not a discrete set but a continuous range, described by the angle between the fingertip and

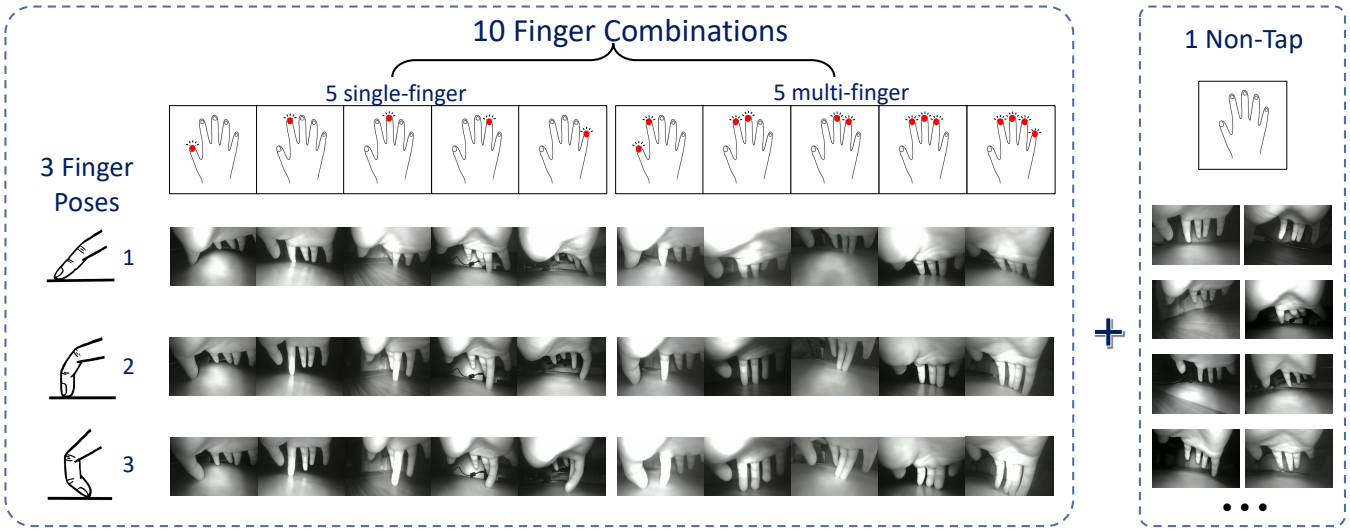


Figure 2: Training gesture set. A total of 31 categories, including 30 tapping gestures composed of 10 finger combinations and 3 finger postures, plus 1 non-tap gesture. Image samples corresponding to different finger combinations from various users are displayed as well.

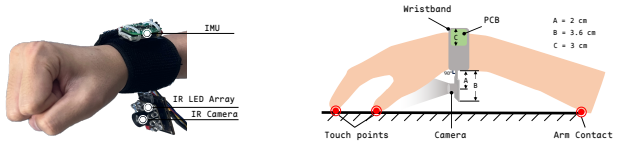
the surface, corresponding to $< 90°$, $\approx 90°$, and $> 90°$, respectively. Choosing fewer postures (e.g. N=1,2) would significantly reduce the number of gesture sets, and more postures would complicate the divisions in the fingertip area, confusing users and leading to errors. Based on these considerations, we chose these three postures.

To answer **Q2**, our goal is for the model to recognize all finger combinations. A straightforward approach would involve collecting data for all 31 combinations, but this would require substantial resources in terms of data collection, participant recruitment, and model training. To address this, we adopt a strategy of using a limited subset of gestures to learn general finger movement representations and infer the distribution of the full gesture set. For other potential finger combinations, we propose a user-adaptive few-shot learning approach to enhance accuracy. This method requires minimal resources, making it a highly efficient solution. This subset of gestures was chosen based on learning theories and ergonomic considerations. Single-finger tapping provides single-label data, offering clear supervisory signals that enhance the model's ability to effectively learn distinguishing features between categories [5]. Finger tapping efficiency and comfort are significantly influenced by the anatomical and neurological connections between fingers. Adjacent finger pairs, such as index-middle or middle-ring, are more ergonomic due to shared tendons and synchronized muscle control, which facilitate coordinated movements. In contrast, non-adjacent pairs like index-ring or middle-little involve reduced motor coordination and higher muscle strain. Studies by Hager-Ross et al. [24] highlight the limited independence of the ring finger, which is prone to involuntary movement, while research by Francis and Kinoshita [3] shows that non-adjacent finger combinations result in slower tapping cadence and greater fatigue. Additionally, Loehr et al. [46] emphasize biomechanical constraints, where the movement of one finger often influences others, particularly less independent ones like the ring finger. Based on this, we chose ten gestures, including five simple single-finger gestures and five multi-finger gestures where the tapping fingers are adjacent.

3.2 Hardware Prototype

One of the key challenges faced by vision-based sensing modalities is interference from ambient light. To mitigate this issue, we use an infrared camera equipped with infrared LEDs. Additionally, a motion sensor (i.e. IMU) capture wrist vibration signals to detect tapping events. Our wristband, as shown in Fig. 3a, features a circular design comprising a Velcro strap, an IMU, an infrared camera, and an array of infrared LEDs. The infrared camera (OV2710) is equipped with a 3.5mm lens, providing a 100-degree field of view. It captures 1920×1080 resolution infrared images at 30 fps, transmitted via USB. Surrounding the camera is an array of 10 infrared LEDs operating at a wavelength of 850nm. The LEDs activate simultaneously with the camera. The ICM-42688-P 6-axis MEMS is integrated into the PCB and powered by an independent lithium battery. Sensor signals are upsampled to a frequency of 100 Hz and transmitted to the host computer via UART. Fig. 3b shows the wristband's form factor: the IMU is mounted on a 3 cm square PCB attached to the strap via Velcro, while the infrared camera is fixed perpendicularly to the wrist using an acrylic mount, positioned 2 cm below the wrist. This design ensures that the infrared LED array illuminates the finger area rather than the palm. During tapping, users can rest their elbow or forearm muscles on the table, naturally raising the wrist for typing. The entire device weighs 36.93 g (excluding cables), lighter than the Apple Watch Series 9 (42.3 g).

Why Fusion Sensors? Although technologies [7, 52, 62] can use one or more IMUs on the wrist to detect specific finger taps, they struggle to exceed 10 gestures while maintaining a high accuracy. Considering our gesture set, which involves not only combinations of fingers but also their postures, IMU signals cannot achieve such fine-grained distinctions. Digits [36] employs an IR



(a) Hardware Prototype (b) Wristband's form factor

Figure 3: *FineType*'s hardware prototype includes a wristband with an IMU fixed above it and an infrared camera with IR LEDs positioned under the wrist.

camera mounted below the wrist, paired with an IR laser projector, to estimate the hand's 3D pose, showcasing the potential of gesture regression from this camera setup. Therefore, capturing images during tapping moments using the camera was chosen as input for the model. Nevertheless, keeping the camera active at all times and performing inference on every frame to determine the tapping gesture would result in high power consumption, making it impractical for user applications. In contrast, using an IMU to detect tapping events offers robust, low-latency, and low-power contact detection compared to using optical sensors [22]. Hence, we decided to integrate both IMUs and cameras into our system.

3.3 System Overview

Fig. 4 illustrates the overall system flow of *FineType*. We use an IMU sensor to detect the distinct wrist vibrations caused by finger taps. When detected immediately, we activate the camera to capture images at the moment of tapping. We have developed a set of algorithms for fine-grained finger gesture recognition, employing a multi-task CNN to determine the combinations and postures of fingers tapping the surface. This process already supports user interactions such as text entry and mapping function keys. For users who wish to further enhance their accuracy with specific finger combinations, we have designed a user-adaptive transfer learning method, capable of mapping all finger combinations. *FineType* utilizes standard commercial sensors, making it suitable for integration into existing wristband devices or as a standalone wearable for daily life and XR applications in text input and command control.

4 *FINETYPE* MODEL

4.1 Tap Image Capture

When a user taps the surface, the vibrations from the finger locations propagate through the hand and are precisely captured by the 3-axis accelerometer at the wrist, defined as the acceleration vector $a(t) = [a_x(t), a_y(t), a_z(t)]$. Based on our observations, a tapping event typically causes a noticeable change in the accelerometer signal lasting approximately 200-400 ms, during which the finger remains in contact with the surface for about 3-5 frames. We need to detect the tap event as the finger contacts the surface and then activate the camera to capture the frame. The finger tap causes a strong, short-duration change in the accelerometer, characterized by a significant spike in the acceleration magnitude $\|a(t)\|_2$. Due to sensor noise and numerical errors, We use a moving average filter to smooth the signal, with a window size of $N = 20$, defining the average energy at a given moment as

$$E(t) = \frac{1}{N}\sum_{i=0}^{N-1}[a_x(t-i)^2 + a_y(t-i)^2 + a_z(t-i)^2], \quad (1)$$

and we take the first-order difference of $E(t)$ yields the abrupt signal $I(t) = E(t) - E(t-1)$. We then traverse the data to find local maximum, which indicate the moment just a finger taps the surface. Let t_l and t_r represent the local minima of the signal I on the left and right sides of time t, respectively. The prominence of the signal $I(t)$ is defined as:

$$P(t) = I(t) - \min\{I(t_l), I(t_r)\}. \quad (2)$$

To prevent sensor drift errors from repeatedly triggering tap events, we consider a tap event valid only when the prominence $P \geq 0.06$. Furthermore, after detecting a valid peak, a 200 ms detection pause is introduced, allowing up to 300 input events to be detected per minute. Since the signal is collected in real time, if the current time t is a tapping moment, it takes approximately 3–5 additional accelerometer data points to compute the prominence. Once a valid peak is detected, the camera is triggered to capture the frame. Fig. 5 illustrates the signal variations observed while inputting the string "finetype" using tapping gestures. We also tested the tap detection method from TapID [52] and found it typically detects taps 50~100 ms earlier than our method, often capturing the finger before it touches the surface. This is likely because TapID captures moments of rapid acceleration signal changes, which do not necessarily indicate contact with the surface. In contrast, our method detects the inflection point of the average energy, which only occurs when the finger tap is impeded by the surface, causing a sudden energy drop.

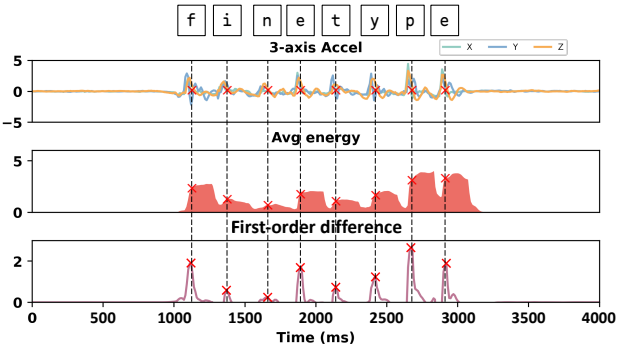


Figure 5: Tap detection diagram. Our algorithm captures the moment with the highest energy change from a 3-axis accelerometer to trigger the camera and capture the tapping moment image. The dashed lines and red cross mark the tapping timestamps.

4.2 Data Collection

4.2.1 Participants and Apparatus. We recruited 28 volunteers (24 males, 4 females), aged 18–33 years (Mean = 23.6, SD = 3.2), all right-handed. Each participant spent ~30 minutes on data collection and received a $15 reward. Our hardware setup (Fig. 6a) consisted of an IMU transmitting data to a laptop at 100 Hz via UART and an

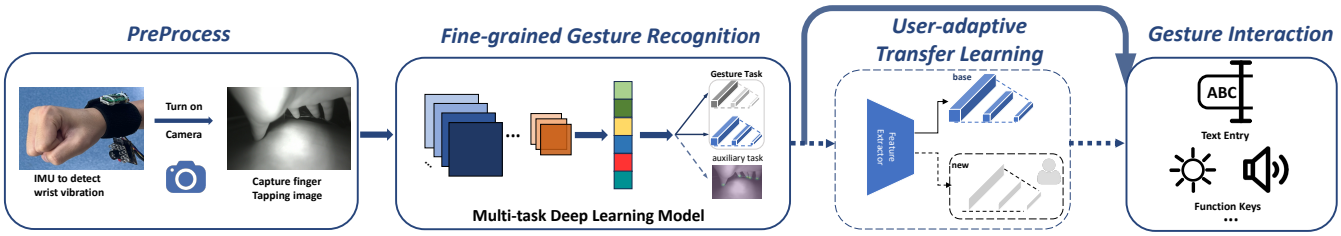


Figure 4: System overview of *FineType*, consisting of four main modules: Tap event preprocessing, Fine-grained gesture recognition, User-adaptive transfer learning for gestures (optional), and Interaction with tapping gestures.

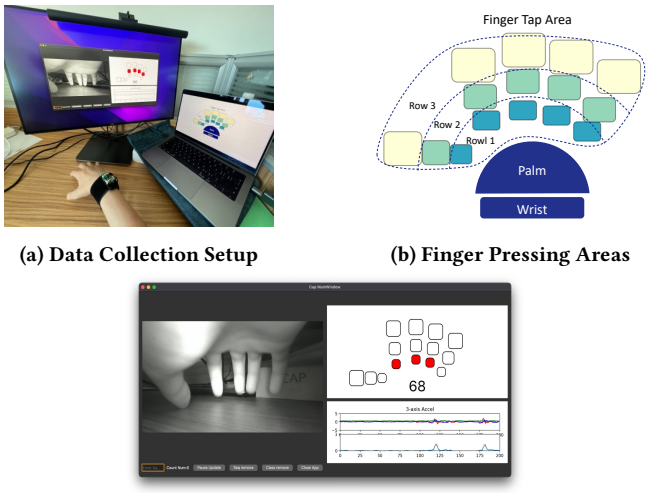
infrared camera connected via USB. Upon detecting a valid tap (Section 4.1), the camera captured 8 consecutive frames at 30 Hz.

4.2.2 Gesture Data. To clearly define the gesture set for users, we provided printed diagrams of finger pressing areas (Fig. 6b). During typing, the wrist and palm remained stationary, allowing free movement of the fingers. Users were instructed to press within 3 predefined finger areas (Rows 1, 2, and 3) using 3 postures and 10 finger combinations to ensure clarity. After sufficient practice and confirmation of user understanding, formal gesture collection began. Each finger combination and posture (Section 3.1) was repeated four times, resulting in $4 \times 30 = 120$ taps. The gesture collection interface (Fig. 6c) displayed the required finger combination and posture in red, alongside real-time IMU signals and tap images captured by the camera. Before each tap, the interface highlighted the upcoming gesture; upon tap detection and image capture completion, it automatically advanced to the next gesture. To minimize order effects, gestures were arranged using a Latin-square design.

We accounted for varying lighting conditions during data collection, with 13 participants recorded under indoor night lighting, 8 outdoors in dim evening light, and 7 in well-lit daytime settings. The dataset also includes instances of motion blur caused by rapid tapping, low light intensity, partial finger occlusions, and slight image rotations, ensuring coverage of diverse real-world scenarios.

4.2.3 Non-tap Data. In real-world scenarios, various movements, such as finger pinching or wrist shaking, can produce signals similar to faint finger taps, making them difficult to differentiate using only an IMU. To address this, our system analyzes images to verify whether wrist vibrations result from finger taps. During gesture data collection, we captured eight frames starting from the tap, including many instances where no surface contact occurred. Instead of collecting additional non-tap data, we extracted non-contact instances from the last frame of the eight-frame sequences. We randomly selected the last frame from 300 gestures in the dataset as Non-Tap data. Since these gestures do not involve surface contact, finger postures are unrestricted and not further categorized.

4.2.4 Annotation. Our program automatically annotates images collected during data acquisition, requiring only manual annotation of fingertip positions. Common finger keypoint detection algorithms and devices (e.g., MediaPipe[1] and LeapMotion[2]) often



(a) Data Collection Setup **(b) Finger Pressing Areas**

(c) Capture Demo

Figure 6: Training gesture collection diagram. Users are instructed to perform actions as indicated by the capture program (the red square indicates the finger gesture to be pressed). Then the program automatically records IMU and image data at the moment of tapping and annotates it.

exhibit inaccuracies or fail under our camera setup. To ensure precision, we manually annotated the five fingertip positions in each image. Incorrect tap samples were excluded, resulting in a final dataset of 3,541 images.

4.3 Model Development

4.3.1 Multi-Task Network. Multitask Learning (MTL) leverages shared information across related tasks to improve neural network generalization, enhancing performance, efficiency, memory usage, and convergence speed [67]. The primary task of our model is to recognize finger postures during surface taps and identify simultaneously pressed fingers. Given the significant variation in fingertip regions with different postures—where more bent fingers make fingertips more prominent—we incorporate a secondary task to predict fingertip heatmaps. This secondary task widely used in keypoint detection is designed to help the model focus on finger contact features, reducing the likelihood of overfitting to irrelevant environmental details.

[1]https://mediapipe-studio.webapps.google.com/demo/hand_landmarker
[2]https://www.ultraleap.com/

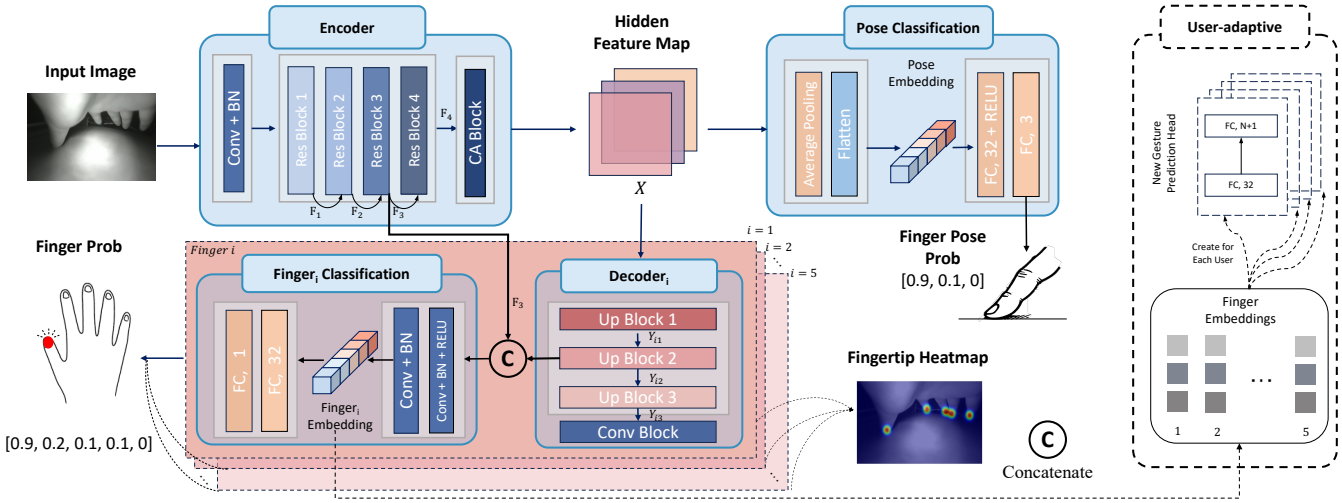


Figure 7: A multi-task multi-label CNN for fine-grained tapping gesture recognition; along with an optional user-defined gesture classification network.

4.3.2 Multi-label Finger Representation. To enable the model to predict all possible finger combinations, rather than just the predefined ones in 3.1, we reframed the problem as a multi-label classification. Each finger is treated as a separate label, marked as 1 if tapped and 0 if not. This approach allows the model to learn the unique characteristics of each finger when pressed, enabling recognition of all finger combinations.

4.3.3 Model Architecture. We developed an encoder-decoder network architecture, as shown in Fig. 7. The encoder starts with initial feature extraction using 64 convolutional kernels (3×3) combined with Batch Normalization (BN), followed by four residual convolutional blocks from the ResNet18 [26] backbone, each using 3×3 kernels. The number of kernels increases from 64 to 128, 256, and 512, yielding feature maps F_1, F_2, F_3, F_4. We utilized pretrained ResNet18 weights from the ImageNet dataset [8] to accelerate training convergence. Following convolution, a Coordinate Attention (CA) module [29] captures spatial location information, enhancing the network's understanding of spatial relationships and inter-channel dependencies. The feature map F_5 output by the CA module undergoes global average pooling and is flattened into a 512-dimensional latent vector X, which is then processed by two MLP layers for posture classification. For the heatmaps of each finger, we constructed five identical decoder modules. Each decoder contains three upsampling convolution layers, doubling the feature map dimensions, with channel counts of 128, 64, and 32, followed by a convolution layer to output a 56×56 regression heatmap. During upsampling, each input feature is concatenated with the corresponding encoder feature map F_i (for $i = 1, 2, 3, 4, 5$), resulting in output feature maps $Y_{i1}, Y_{i2}, Y_{i3}, Y_{i4}$.

To improve the network's focus on fingertip positions when predicting tapping finger combinations, we combined deep feature information from both the image and the fingertip heatmap regression. For the ith finger, we concatenated F_3 with Y_{i2} and passed it through a two-layer convolutional module with 3×3 kernels (16 and 4 filters, respectively) and BN. This was followed by an MLP classification head that outputs the sigmoid value for that finger.

4.3.4 User-adaptive New Gesture Recognition. While our multi-label tagging approach enables the model to recognize all finger combinations, the absence of unseen combinations in the training set may lead to misclassification of new gestures as existing categories. However, our network, trained on extensive gesture data, has developed robust general finger representation features. We propose a user-adaptive few-shot method aimed at leveraging these general features to quickly adapt to new gesture categories. With just a few simple registrations of new gestures, users can achieve accuracy that surpasses the original performance. In this approach, we freeze the feature extraction layers trained on the original gesture set and introduce a new lightweight classification head for each user, as shown in the right block diagram of Fig. 7. By training only this fully connected prediction head while keeping the feature extractor weights fixed, the method eliminates the need for backpropagation through the entire network, significantly reducing memory usage and achieving training speeds over two orders of magnitude faster than conventional fine-tuning methods, as shown in Table 5. This process also removes the requirement for annotated finger heatmaps, making it more suitable for commercial deployment. By leveraging a small number of samples for new gestures, this method extends the network's adaptability without necessitating complete retraining, providing a practical and efficient solution. For implementation details, refer to Appendix A.

4.4 Model Training

4.4.1 Data Augmentation. Our analysis of the training data reveals that users exhibit slight variations in wrist tilt, either left or right, when performing gestures, and their wrist heights relative to the desk also differ. Some users keep their wrists close to the desk for comfort, while others raise them to facilitate finger movement, which impacts both the camera angle and the infrared light path.

Additionally, user habits lead to significant differences in image brightness and contrast, even for the same gestures. Images captured in daylight often display lower hand contrast and higher background brightness. The wristband's positioning also varies, with some users wearing it closer to the hand, making it more visible in the images, while others wear it further away, capturing more of the hand. To address these variations and prevent overfitting, we propose five data augmentation techniques: 1) image flipping to account for both hands, 2) brightness adjustment using a scaling factor drawn from a uniform distribution $S_{\text{bright}} \sim U(s_{\min}, s_{\max})$, with $s_{\min} = 0.5$ and $s_{\max} = 1.5$; 3) contrast adjustment using a similar scaling factor $S_{\text{contrast}} \sim U(s_{\min}, s_{\max})$; 4) random cropping of a 200×200 block from the input image followed by resizing to the original resolution to simulate variations in camera position and angle; and 5) random rotation within a range of -10 to 10 degrees to model slight wrist tilts.

4.4.2 Loss Function. Our multi-task network mainly includes three tasks: multi-label finger classification, finger posture classification, and fingertip position heatmap regression. For the multi-label finger classification, the loss $\mathcal{L}_{\text{finger}}$ is calculated by determining the binary cross-entropy loss $\mathcal{L}_{\text{BCE}}^{(i)}$ for each finger i, as follows:

$$\mathcal{L}_{\text{finger}} = \frac{1}{5} \sum_{i=0}^{5} \mathcal{L}_{\text{BCE}}^{(i)}. \tag{3}$$

For the finger posture classification loss $\mathcal{L}_{\text{pose}}$, we use cross-entropy loss. However, for non-tap images that do not make contact, posture information cannot be determined. Thus, we define the posture loss as:

$$\mathcal{L}_{\text{pose}} = \frac{1}{N - N_{\text{non}}} \sum_{i=0}^{N} \mathbb{I}(i) \cdot L_{\text{CE}}(\tilde{p}, p), \tag{4}$$

where N represents the total number of data points, N_{non} represents the number of non-tap data points, $\mathbb{I}(i)$ is 0 if the ith data point is non-tap and 1 otherwise, and $\tilde{p}, p$ represent the softmax output of the probability prediction and the true label of the finger posture, respectively. For fingertip heatmap regression, we use the MSE loss function, defined as $\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{MSE}}(\tilde{Y}, Y)$, where $\tilde{Y}, Y$ represent the predicted and actual values of the fingertip heatmap regression, respectively. Thus, our overall loss function can be expressed as:

$$\mathcal{L} = \alpha_{\text{finger}}\mathcal{L}_{\text{finger}} + \alpha_{\text{pose}}\mathcal{L}_{\text{pose}} + \alpha_{\text{reg}}\mathcal{L}_{\text{reg}}, \tag{5}$$

where the weights $\alpha_{\text{finger}}, \alpha_{\text{pose}}, \alpha_{\text{reg}}$ are used to balance the importance of these three tasks. We expect that the fingertip regression task helps the model focus better on the finger tapping details within the image, and this task is as important as the other tasks. Therefore, we use dynamic weight adjustment, selecting the reciprocals of the respective loss functions for $\alpha_{\text{finger}}, \alpha_{\text{pose}}, \alpha_{\text{reg}}$ in each iteration, ensuring that the importance of the three tasks is consistent during each backpropagation.

4.4.3 Training Settings. To evaluate model performance and mitigate overfitting, we utilize leave-one-out cross-user validation. Images are resized from 1920×1080 to 224×224 to match the network input requirements. The heatmaps are computed by applying a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ centered at the annotated fingertip coordinates, with $\sigma = 5$. Training is performed using the Adam optimizer with an initial learning rate of 0.001, which is dynamically adjusted via the ReduceLROnPlateau strategy over 200 epochs. All experiments are conducted on an NVIDIA GeForce RTX 3090 GPU with 24GB VRAM.

5 EVALUATION

We will validate our model's recognition capabilities on both the predefined gesture set and a new set of gestures to demonstrate its generalizability. Additionally, we conducted ablation studys to prove the advantages of our method.

5.1 Predefined Gestures Detection

5.1.1 Cross-user Validation. We utilized a leave-one-out cross-user validation to evaluate our model. The dataset from N users, described in Section 4.2, was divided into N folds $[P_1, P_2, \ldots, P_N]$, with each fold using data from $(N-1)$ users for training and the remaining user for testing. This cross-user validation method, through iterative training and testing, provides a more accurate assessment of the model's generalizability while minimizing overfitting risks. The model's performance will be evaluated from two key perspectives:

- **Gesture-Level**: We evaluate the model's performance from a gesture classification perspective, assessing its ability to recognize finger combinations (11 categories: 10 finger combinations + 1 non-tap), finger postures (3 categories), and fine-grained gestures (31 categories: 30 gestures + 1 non-tap). For each evaluation mode, we compute the average accuracy and Macro-F1 score across all categories. Additionally, for finger combination and all fine-grained gestures, we calculate the false positive rate for the non-tap category.
- **Finger-Level**: We examined the model's classification performance for individual fingers. For each finger, we computed the accuracy and false positive rate for binary labels, to assess the model's ability to recognize different fingers.

The final evaluation metric is the average of all users' validation results (accuracy, F1 score, and false positive rate). For any validation metric E, given user evaluations $[E_1, E_2, \ldots, E_N]$ $(N = 28)$, the result is calculated as $E = \frac{1}{N}\sum_{i=1}^{N} E_i$.

5.1.2 Ablation Study. We remove data augmentation method (Section 4.4.1) and the fingertip regression auxiliary task (Section 4.3.1) individually to evaluate their respective contributions to the results.

5.1.3 Results. Table 2 summarizes the evaluation metrics for gesture and finger recognition from two ablation studies, compared to our baseline method. Our approach achieved accuracies of 98.26%, 95.53%, and 94.19% for finger combinations, posture classification, and overall gesture recognition, respectively, with corresponding F1-scores of 98.45%, 95.53%, and 93.84%. These results demonstrate significant improvements over methods without data augmentation or the auxiliary task. Moreover, our method reduced the false positive rate (FP Rate) for non-tap gestures to 1.09%, compared to 2.48% without data augmentation and 2.66% without the auxiliary task. At the finger level, our method consistently outperformed others for the index, middle, ring, and little fingers. These findings confirm that our approach effectively reduces false positives

Table 2: The average accuracy (%), F1-score (%), and FP Rate (%) at the Gesture-Level and Finger-Level for various ablation experiments and our method.

Metric	Method	Gesture-Level			Finger-Level				
		Combination	Posture	Overall	Thumb	Index	Middle	Ring	Little
Acc	w/o Augmentation	97.58	93.17	91.24	99.74	99.62	98.95	99.01	99.15
	w/o Auxiliary Task	98.18	94.58	93.04	99.70	99.55	99.12	99.23	99.65
	Ours	98.26	95.53	94.19	99.66	99.66	99.17	99.71	99.77
F1-score	w/o Augmentation	98.09	93.16	90.59	99.33	99.58	98.88	98.72	97.88
	w/o Auxiliary Task	98.39	94.57	92.31	99.26	99.52	99.08	98.95	99.03
	Ours	98.45	95.53	93.84	99.09	99.63	99.11	99.62	99.33
FP Rate	w/o Augmentation	2.48	-	2.48	0.26	0.26	0.80	0.75	0.61
	w/o Auxiliary Task	2.66	-	2.66	0.30	0.21	0.45	0.38	0.10
	Ours	1.09	-	1.09	0.29	0.35	0.49	0.24	0.03

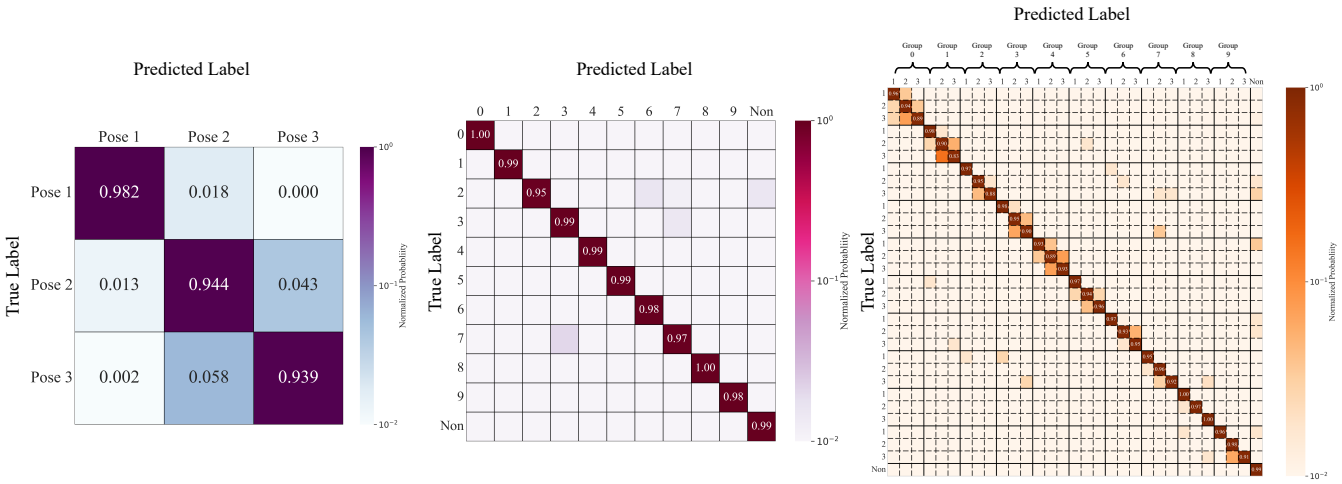


Figure 8: Gesture recognition confusion matrix: From left to right, the figures represent three finger postures, 10 finger tapping combinations plus 1 non-tap, and all fine-grained gesture categories, with average accuracies of 98.26%, 95.53%, and 94.19%, respectively.

while enhancing the accuracy and reliability of gesture and finger recognition.

The three confusion matrices in Fig. 8 illustrate the results of finger pose classification, finger combination classification, and fine-grained gesture classification. From the finger pose classification results (left), the model performs exceptionally well across the three pose categories: Pose 1 (middle fingertip touch) achieves an accuracy of 0.982, Pose 2 (front fingertip touch) an accuracy of 0.944, and Pose 3 (fingernail touch) an accuracy of 0.939. However, the confusion rate between Pose 2 and Pose 3 reaches 0.058, indicating a certain degree of ambiguity between these two poses. In the finger combination classification results (middle), all labels achieve an accuracy greater than 0.97, except for label 2 (0.95). The results of fine-grained gesture classification (right) further demonstrate that confusion primarily occurs between different finger poses within the same finger combination. Overall, the model performs well in finger combination classification, but as task complexity increases with the introduction of finer-grained poses, the classification accuracy decreases. This is particularly evident in the confusion between different finger poses within the same gesture, especially between Pose 2 and Pose 3.

5.2 Newly Defined Gestures Detection

5.2.1 New Tapping Gesture Set. We introduced 24 new gestures, consisting of 8 finger combinations paired with 3 finger postures, as illustrated in Fig. 9. These gestures were not included in the training data and were designed to evaluate the model's generalization capability, specifically its ability to recognize unseen finger combinations. The gestures encompass four categories of multi-finger combinations, ranging from two- to five-finger taps, prioritizing the flexibility of the thumb and index finger (6 and 5 instances, respectively) while reducing reliance on the ring and pinky fingers (4 and 3 instances, respectively) to improve tapping comfort [3].

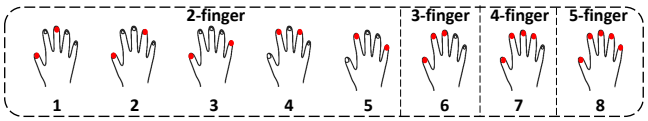


Figure 9: Newly defined gesture set not present in the training data, composed of 8 new finger combinations and their corresponding 3 finger postures.

5.2.2 Data Collection. Eight new volunteers (6 males, 2 females, Age = 23.1 ± 2.7) participated in our data collection study, including one left-handed individual. Using the program described in Section 4.2, we adjusted the gestures being collected. Each participant performed each gesture 6 times, resulting in 18 gestures per finger combination (3 postures × 6 repetitions) and a total of 144 gestures (8 categories × 18 samples). The collection followed a Latin square order, ensuring that each gesture appeared only once in each set of 24 collections, minimizing order effects. In total, 1152 test samples were collected (8 participants × 144 samples).

5.2.3 Model Performance. The new gesture set was used exclusively as the test set, without any labeling or data augmentation. Training data from Section 4.2 were randomly split into training and validation sets at a 9:1 ratio, ensuring diverse user data and characteristics. The model achieving the highest gesture recognition accuracy on the validation set was selected for testing on the new dataset, which was not included in training. We conducted ablation experiments following the setup in Section 5.1.2 and evaluated the new gestures under 3-shot and 6-shot settings using the method from Section 4.3.4. Since few-shot learning requires half of the new gesture set for training and validation, the remaining half was used for testing. Table 3 summarizes the average accuracy across various finger combinations and postures under different settings. Our method achieved an average accuracy of 91.27% across the 8 new finger combinations, significantly outperforming results without data augmentation (82.64%) and without auxiliary tasks (14.88%). This highlights the impact of auxiliary tasks in directing the model's attention to finger regions, improving accuracy for both finger combinations and postures by focusing on finger-related features. Additionally, the 3-shot and 6-shot methods improved accuracy from 89.06% to 92.71% and 97.05%, respectively, demonstrating the model's ability to recognize all potential finger-tapping combinations with minimal user registration. As users provide more gesture registrations within the same category, classification performance further improves, reinforcing the adaptability and scalability of our method.

Table 3: Average accuracy (%) of ablation experiments, few-shot methods, and our method across 8 new finger combinations and 3 finger postures. "Aug" represents data augmentation, and "Aux" represents auxiliary tasks.

Condition	w/o Aug	w/o Aux	Ours	Ours*	3-shot*	6-shot*
Combination	82.64	14.88	91.27	89.06	92.71	97.05
Posture	86.61	91.27	93.86	94.44	94.44	94.44

* indicates the method is tested on remaining set (i.e. 50% of new set).

6 TEXT ENTRY USER STUDY

6.1 Study Design

6.1.1 Participants. We recruited 10 participants for our user experiment through social media, with ages ranging from 18 to 29 years (Mean = 23.9, SD = 3.3). The group included 7 males and 3 females, all of whom had experience using QWERTY keyboards (Mean = 10.8, SD = 4.4). All participants were right-handed. Each participant received a base stipend of $30. Additionally, participants who achieved a character error rate of less than 5% and a typing speed above 35 WPM were given an extra tip as an incentive.

6.1.2 Apparatus. Participants were instructed to wear the *FineType* wristband comfortably and sit in front of a monitor. The wristband device transmitted IMU signals and images to a backend PC via UART and USB, respectively, similar to the data collection setup. The gesture recognition model, implemented in PyTorch, was deployed on a MacBook M2 Pro (16GB memory) connected to a power source. Using the finger tapping detection method described in Section 4.1, the system detects tapping gestures to trigger the camera and immediately performs inference with the recognition model. The model achieved an average inference speed of 27 ms over 1,000 runs. The testing program was designed based on the TextTest++ [80] platform, logging inference results with timestamps as local files to enable subsequent analysis.

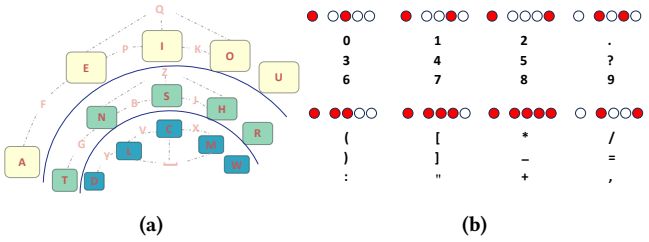
6.1.3 Task. We assessed the text entry performance of the *FineType* system through two within-subject tasks:

- **Task 1: Normal Phrase Set.** This task evaluates the model's ability to input text consisting of the 26 English letters. All target phrases are randomly selected from MacKenzie and Soukoreff's phrase set [47]. We compare our method with TapXR [34], which requires users to memorize distinct finger combinations for character input. Additionally, we record typing performance using a standard keyboard and one-handed input on a smartphone touchscreen for comparison.
- **Task 2: A more comprehensive character set.** This task examines the performance differences between *FineType* and TapXR on a broader set of phrases (letters, numbers, and symbols). We randomly selected 7 strings from the Enron Email Set [38, 68], which consists of real-world email content, and 8 strings from CodeSearchNet [31], a dataset of code snippets in various programming languages, forming a total of 15 strings. The average string length is 11.2, with letters, numbers, and symbols accounting for 41.2%, 18.1%, and 40.7% of the characters, respectively. All symbols shown in Fig. 10b are included in the test set.

6.1.4 Keyboard Layout. Our gesture system allows flexible mapping to any letters, enabling users to customize layouts based on their preferences and comfort. For the user study, we provided a single keyboard layout, as shown in Fig. 10a. Using tapping gestures with '○ ● ● ● ●', we assigned 'Delete,' 'Enter,' and 'Shift' to the three postures, respectively. The layout prioritizes the comfort of single-finger taps by mapping the most frequently used letters to five single-finger gestures across three postures. Remaining letters were assigned to two- and three-finger gestures, arranged by letter frequency [74] from high to low. Vowels ('a, e, i, o, u') were positioned on the keyboard's outermost layer, reflecting their unique

linguistic importance. Fig. 10b demonstrates the use of 8 new gestures across 3 postures to map numbers and symbols (excluding letters) in Task 2. Unlike TapXR, our system eliminates the need for multiple taps or numeric keyboard switching.

Why Differs from QWERTY Layout? Our typing system relies solely on one hand. For right-handed users, the QWERTY layout assigns only about 11 letters to the right hand, leaving 15 letters that require learning multi-finger gesture mappings[3]. On the other hand, based on English typing frequency [74], our layout achieves an average finger-tap count[4] of 1.12, compared to 1.93 for the QWERTY layout. This indicates that, during long-term text input, our layout requires about 42% fewer finger taps than QWERTY, offering a more relaxed and comfortable typing experience. Given that the QWERTY layout still requires learning and involves more finger taps, we opted to use our proposed keyboard mapping for each participant in the user study.



(a) (b)

Figure 10: *FineType* keyboard layout. (a) Letter mapping: Dashed lines indicate that the character requires simultaneous pressing of multiple fingers connected by the dashed lines. (b) Non-letter characters mapping.

6.2 Procedure

The user study was conducted for all participants at a pre-arranged time in a quiet conference room. None of the participants had prior experience with *FineType* gestures or TapXR.

Task 1 involved 4 text input test sessions: *FineType*, TapXR, physical keyboard, and single-hand touchscreen typing. The testing was split into 2 phases, spaced 7 days apart, to minimize the influence of memorizing different gesture-to-letter mappings. Phase 1 tested *FineType*, the physical keyboard, and single-hand touchscreen typing, while Phase 2 tested TapXR. Each session consisted of 5 text input blocks, with each block containing 5 randomly selected phrases from the MacKenzie and Soukoreff's phrase set [47]. The same phrases was used for all participants but was randomly reordered for each individual. Phrases did not repeat within a session and were consistent across sessions.

Participants underwent a structured training process to familiarize themselves with *FineType*. Training began with a 10-minute explanation of the gesture mapping to ensure a clear understanding of the gestures. Subsequently, participants practiced one-handed input using *FineType* with phrases randomly selected from the remaining MacKenzie and Soukoreff phrase set. During practice, a reference mapping of gestures to keyboard characters was displayed, and participants were allowed up to two hours to practice. The testing phase began once participants confirmed that they had fully memorized the gesture-to-character mappings. Participants were instructed to type as quickly and accurately as possible without pausing during a session. If a pause occurred during transcription, the phrase had to be retyped from the start. Additionally, the keyboard layout will be clearly displayed to assist participants in typing more efficiently. They were allowed to take up to 2 minutes of rest between phrases and up to 5 minutes between blocks. Testing also included baseline comparisons with keyboard typing and single-hand touchscreen typing. In the second phase, TapXR was evaluated following the same procedure. Participants were informed that the system would start timing once the first letter was entered. Therefore, they could memorize the target phrase before starting to type if they wished. Four months after Task 1, the same participants were invited to participate in Task 2. Task 2 included two text entry testing sessions comparing *FineType* and TapXR, each consisting of 3 blocks with 5 strings per block. All participants received the same set of strings, with the order randomized within each block. Unlike Task 1, a 6-shot learning approach was applied before the practice phase to train a classification head for the 8 new finger combinations with 3 postures, which operated alongside the original model. Participants practiced for up to two hours before beginning the test sessions. All other testing settings remained consistent with Task 1.

6.3 Results

We evaluated text entry speed using Words Per Minute (WPM) [4], calculated as the time elapsed between entering the first and last characters of the transcribed text. Error rates were assessed with three metrics from TextTest++ [61, 81]: uncorrected error rate (UER), corrected error rate (CER), and total error rate (TER). Additionally, we calculated the overall character error rate (ChER[5]), defined as the Levenshtein distance between the transcribed and reference texts, divided by the length of the reference text [62, 69]. ChER is similar to UER but disregards corrections in its calculation.

6.3.1 Text Entry Performance. We evaluated typing performance across two tasks, measuring typing speed in WPM and error rates (UER, CER, TER, and ChER), as shown in Fig. 11. In Task 1, participants achieved an average of 74.4 WPM on a physical keyboard, with a ChER of 0.3%, UER of 0.3%, CER of 2.9%, and TER of 3.2%. One-handed touchscreen typing averaged 37.6 WPM, with an average ChER of 0.7%, UER of 0.5%, CER of 4.7%, and TER of 5.2%. *FineType* reached an average of 35.1 WPM with a ChER of 5.1%, UER of 4.6%, CER of 5.3%, and TER of 9.9%. TapXR achieved an average of 29.5 WPM with a ChER of 3.5%, UER of 3.2%, CER of 3.6%, and TER of 6.8%. *FineType* reached 93% of the speed of one-handed touchscreen typing. In Task 2, *FineType* achieved an average of 15.0 WPM with a ChER of 7.5%, UER of 6.9%, CER of 5.8%, and TER of 12.7%, while TapXR reached 9.9 WPM with a ChER of 22.9%, UER of 16.6%, CER of 7.5%, and TER of 24.1%. Participants were instructed

[3]Single-finger gestures map to columns starting with 'y, u, i, o, p,' while multi-finger gestures map to columns starting with 'q, w, e, r, t.'
[4]Calculated by multiplying the letter frequency by the number of fingers required for the corresponding gesture and summing the results.

[5]Following TouchInsight [63], we abbreviate character error rate as ChER to distinguish it from corrected error rate (CER).

Table 4: The means and standard errors of WPM, UER (%), CER (%), and TER (%) for different blocks in Task 1 and Task 2 across method (*FineType*, TapXR). '↑' indicates higher is better, while '↓' indicates lower is better.

Metric	Method	Task 1					Task 2		
		Block 1	Block 2	Block 3	Block 4	Block 5	Block1	Block2	Block3
WPM ↑	TapXR	30.6 (1.7)	28.4 (1.0)	30.5 (1.9)	28.2 (1.3)	29.6 (1.4)	9.8 (0.5)	11.3 (0.5)	8.7 (0.4)
	FineType	34.6 (2.5)	34.5 (1.9)	35.2 (2.3)	34.9 (2.1)	36.2 (1.8)	14.6 (1.3)	14.8 (1.2)	15.7 (0.9)
UER (%) ↓	TapXR	3.9 (1.1)	2.9 (0.8)	4.1 (1.0)	2.0 (1.5)	2.9 (1.2)	11.1 (2.2)	18.1 (1.6)	20.6 (2.9)
	FineType	5.7 (1.4)	6.1 (1.2)	5.1 (1.9)	3.5 (1.6)	2.8 (1.2)	9.3 (1.0)	6.5 (1.0)	4.8 (0.9)
CER (%) ↓	TapXR	4.9 (1.1)	3.4 (0.9)	4.8 (1.2)	1.5 (1.0)	3.6 (1.1)	6.3 (0.4)	6.9 (0.2)	9.3 (0.3)
	FineType	6.9 (1.3)	6.4 (1.1)	4.6 (1.6)	3.6 (1.2)	4.9 (1.5)	6.9 (0.3)	5.7 (0.7)	4.9 (0.7)
TER (%) ↓	TapXR	8.8 (1.8)	6.3 (1.7)	8.9 (2.1)	3.6 (2.5)	6.5 (2.2)	17.4 (2.3)	25.1 (1.6)	29.9 (2.9)
	FineType	12.6 (2.7)	12.5 (2.2)	9.7 (3.5)	7.1 (2.6)	7.7 (2.5)	16.2 (1.2)	12.3 (1.6)	9.7 (1.7)

to type as quickly as possible, leaving some errors uncorrected to avoid impacting typing speed [81, 82].

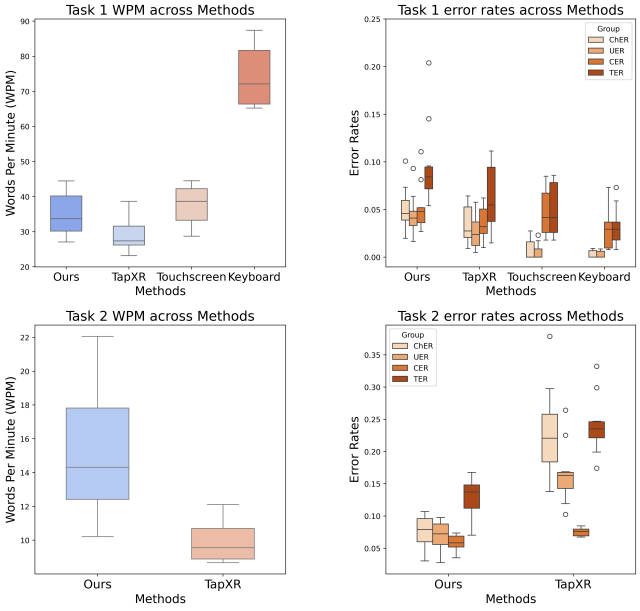


Figure 11: Boxplot of the mean WPM and error rates (ChER, UER, CER, TER) across participants in Task 1 and Task 2.

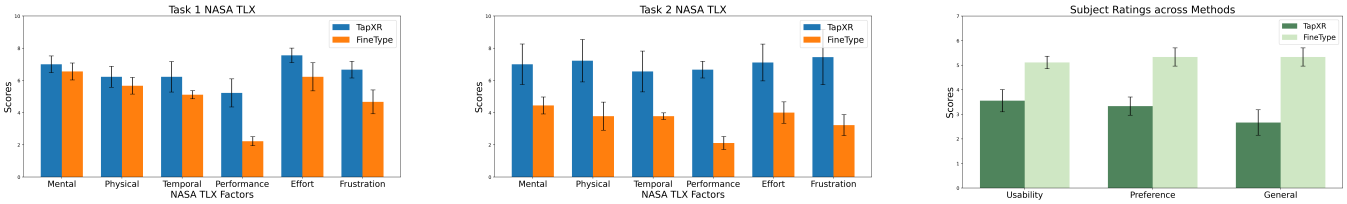
Table 4 presents the average performance across participants for all methods and blocks. To assess differences in various metrics between *FineType* and TapXR, we conducted a two-factor Aligned Rank Transform (ART) ANOVA for Task 1 UER, CER, TER, and Task 2 UER, TER (as the data did not meet normality assumptions). For Task 1/2 WPM and Task 2 CER ($p > .05$ in the Shapiro-Wilk test), standard two-way RM-ANOVA was applied. In Task 1, a significant difference was found between *FineType* and TapXR in WPM ($F_{1,9} = 5.46, p < .05$), but no significant differences were observed in UER, CER, or TER ($F_{1,90} = 1.58, 3.84, 1.87$, all $p > .05$). In Task 2, significant differences were found between *FineType* and TapXR in WPM ($F_{1,9} = 17.94, p < .01$), UER ($F_{1,54} = 58.52, p < .001$), CER ($F_{1,9} = 13.03, p < .01$), and TER ($F_{1,54} = 65.94, p < .001$). The effect of blocks on all metrics was not significant, likely due to higher variability in TapXR's performance across blocks, whereas *FineType* showed more stable performance.

To further investigate whether different metrics for each method vary across blocks, we used Friedman tests for non-normally distributed metrics and one-way RM-ANOVA with Greenhouse-Geisser correction for normally distributed metrics. Post-hoc comparisons were performed using paired sample t-tests with Bonferroni correction. In Task 1, no significant effects were found across blocks for any metric in either method ($p > .1$). In Task 2, *FineType* showed significant effects of blocks for UER ($p = .014$) and TER ($p = .033$). TapXR exhibited significant block effects across several metrics: WPM ($F_{1.55,13.9} = 14.3, p < .001$), UER ($F_{1.33,12} = 4.59, p < .05$), CER ($F_{1.97,17.71} = 25.3, p < .001$), and TER ($F_{1.3,11.67} = 6.79, p < .05$). Post-hoc tests indicated significant differences between Block 1 and Block 3 ($p < .05$). This is probably because Block 3 contains more symbols compared to Block 1, as shown in Appednix Table 6, requiring more frequent tapping in TapXR.

6.3.2 Subjective Feedback and workload. After completing the user tests, we collected user ratings on a 7-point Likert scale in three areas: 1) Usability (ease of use), 2) Preference (liking for the gesture mapping method), and 3) General experience (overall satisfaction with the method). The results, shown in Fig. 12c, indicate that users preferred our keyboard mapping method over TapXR's finger combination tapping and had a generally better experience. Then, we conducted a Wilcoxon signed-rank test on the NASA-TLX [1] questionnaire for Tasks 1 and 2. For Task 1, significant differences were found in performance ($p < .01$), effort ($p < .05$), and frustration ($p < .05$), but not in mental demand, physical demand, or temporal demand. For Task 2, significant effects were observed across all the metrics. The NASA-TLX results for both tasks are shown in Fig. 12a and 12b. We also conducted a 5-point Likert scale survey to assess users' usability and preferences at both the finger and gesture levels during tapping, as shown in Appendix Fig. 17. The results show that users prefer gestures involving the thumb, index finger, and middle finger, as well as adjacent finger gestures.

6.4 Discussion

In Tasks 1 and 2, after up to two hours of user practice, our method achieved average typing speeds of 35.1 and 15.0 WPM, respectively, outperforming TapXR's 29.5 and 9.9 WPM. While no significant difference in error rates (UER, CER, TER, ChER) was found in Task



(a) Task 1 NASA-TLX **(b) Task 2 NASA-TLX** **(c) 7-Point Likert Scale.**

Figure 12: (a-b) NASA-TLX questionnaire results for Task 1 and Task 2, respectively. (c) Subjective ratings using 7-point Likert scale (higher is better). Error bars represent the standard error.

1, our method exhibited significantly lower error rates than TapXR in Task 2. Task 2 involved more complex character mappings, requiring users to memorize additional symbols, resulting in a notable drop in typing speed (approximately 57% for *FineType* and 66% for TapXR) compared to Task 1. Moreover, TapXR relies on multiple sequential taps for non-letter symbol identification, and errors in sequence result in multiple incorrect characters. The performance differences between *FineType* and TapXR can be attributed to the following factors:

1) *Comfort of Key Design.* Our keyboard maps 15 letters using five single fingers, whereas TapXR can only map 5 letters with a single finger, requiring more finger presses overall. Additionally, our method uses adjacent finger combinations for tapping, which is more comfortable, while TapXR involves pressing with non-adjacent finger combinations. Users commonly reported discomfort with gestures corresponding to letters like V (● ● ○ ● ●), W (● ○ ● ○ ●), and J (● ● ● ○ ●) in TapXR.
2) *Diversity Keys of Gesture Sets.* Our method can support up to 93 gestures, while TapXR can only map using 31 finger combinations. As a result, TapXR requires double or triple taps of the same gesture within a short time to map a single symbol. In contrast, our method can map more symbols with just a single tap using three different finger postures for a specific gesture.

We conducted a more quantitative analysis by calculating the normalized character frequencies across the entirety of 3 datasets: MacKenzie and Soukoreff's phrase set, the Enron Email Dataset, and CodeSearchNet. For each dataset $D = \{D_1, D_2, D_3\}$, we then computed the average number of tapping fingers per character $E(T)$ and the average use of non-adjacent gestures $E(N)$. These were calculated as follows:

$$E_i(T) = \sum_{c \in D_i} p_i(c)T(c), \quad E_i(N) = \sum_{c \in D_i} p_i(c)N(c), \quad i = 1, 2, 3$$

Here, $p_i(c)$ represents the probability of character c in dataset D_i, $T(c)$ is the number of fingers (1-5) required to tap for c, and $N(c)$ indicates whether c uses a non-adjacent gesture (1 if used, 0 for all other gestures). The results are shown in Appendix Fig. 18, where it is evident that *FineType* requires fewer tapping fingers on average and uses fewer non-adjacent gestures. This suggests that *FineType* holds great potential for prolonged use and complex symbol input, aligning with the subjective feedback from users.

The results from the NASA-TLX questionnaire further demonstrate the advantages of our method in practical use. Our method performed better in both tasks with higher performance, less effort, and less frustration. According to user feedback, our gestures also outperformed in terms of usability, preference, and overall experience. One user (male, 23 years old) mentioned, "*I feel less fatigued when using fewer fingers or gestures involving the thumb.*" Another user (female, 24 years old) said, "*It's too difficult for me to lift the ring finger while pressing down the middle and little fingers.*" Another user (male, 26 years old, and a skilled pianist) noted, "*I'm proficient at the piano, so these gestures aren't too difficult, but the downside is the lack of rebound feedback similar to typing on a keyboard.*" These experiences indicate that the gestures used for letter mapping in our method are more user-friendly for beginners, while those who are accustomed to using finger combinations (such as piano enthusiasts) find these gestures manageable. Additionally, we developed an n-gram decoder based on finger posture soft decoding (see supplementary materials). By offline correction of transcribed text, the *FineType* character error rate in Task 1 was reduced to 1.6%.

7 APPLICATION SCENARIOS

FineType, by combining finger combinations with 3 different finger postures, can support up to 93 unique single-tap gestures. Based on this extensive set of gestures, we applied it to a variety of scenarios.

Text Entry on Mobile Devices. *FineType* enables text input by tapping on any flat surface, allowing users to place mobile devices (e.g., smartphones or tablets) on a desk and perform accurate input through desk tapping. This method supports a wide range of gestures for mapping various characters, offering two key benefits: (1) it allows smart devices to hide the on-screen keyboard, freeing up valuable screen space; and (2) it enables simultaneous interaction, allowing one hand to perform touch gestures (e.g., cursor positioning, scrolling) while the other inputs text, improving text editing efficiency. We demonstrated a simple mobile typing setup by transmitting recognition results from the PC to a smartphone in real-time via a TCP connection (Fig. 13a). Additionally, we developed a method for controlling cursor movement using the wrist IMU and provided a pre-trained 5-gram decoder for efficient text input and error correction (see in Supplementary Material).

XR Text Input. *FineType* operates without relying on visual feedback or hand tracking from head-mounted displays, making it versatile for XR applications. Using Unity, we developed an AR environment and transmitted *FineType*'s typing results in real-time to Microsoft HoloLens 2 via TCP (Fig. 13b). In this setup, the keyboard

(a) Phone Demo (b) AR Demo (c) Command Demo

Figure 13: *FineType* supports diverse application scenarios, including: (a) single-handed smartphone text input with a hidden keyboard; (b) text input in AR environments with on-screen prompts for text boxes and keyboard layouts; and (c) replacing shortcut key commands with tapping gestures during text editing.

is rendered in real-time, allowing users to position it freely in the real world using gestures. *FineType* detects text input even beyond the headset's field of view, enabling users to focus on the graphical user interface areas. Additionally, it supports using the arm's surface as an input area, allowing users to type by tapping one hand on the other arm, eliminating the need for a desk. This capability enables flexible text input in spatial computing environments, anytime and anywhere.

Additional Command Set. *FineType* supports an extensive set of commands alongside text input, enabling additional control instruction (Fig. 13c). In various applications, gestures can define different shortcuts, allowing the other hand to operate the application seamlessly without interruption.

8 LIMITATIONS AND FUTURE WORK

FineType is worthy of further exploration, as there are still some limitations in applying our current implementation in real-world applications.

Portability. Currently, our device requires a USB connection to a PC for inference, limiting its standalone functionality. The current design is relatively simple, using only two sensors without built-in computational support. In the future, we aim to make the device wireless and compact, ideally integrating it with a smartwatch. Additionally, we plan to employ techniques like model pruning and compression to adapt our method for small, standalone wearable devices with limited computing resources. The current camera design protrudes outward. In the future, we plan to develop a camera that can be embedded into a smartwatch band and discreetly concealed when not in use.

Scenario Diversity. While our method demonstrates reliable recognition performance in cross-user experiments across various scenarios and lighting conditions, it has not yet been tested on a wider range of contact surfaces due to the limited dataset. In the future, we plan to expand data collection to include diverse real-world lighting conditions and contact surfaces, enhancing the model's recognition capabilities. This effort aims to optimize text input solutions for ubiquitous environments in future spatial computing applications.

Classification Accuracy. While our method has achieved reliable recognition on predefined finger combinations, it still has shortcomings in predicting finger postures. This may be because different users have different understandings of finger postures. On the other hand, for newly defined gestures, although we can recognize them using the original model, the performance is still inferior compared to the user-adaptive method. Moreover, the user-adaptive method also requires a simple user registration process. To improve the recognition capability, a practical method is to expand the scope of data collection (including more users' finger postures and all possible finger combinations).

Memorizing Text Entry. In this study, *FineType* was validated with single-hand input, requiring users to memorize letter mappings across multiple fingers and postures, which may pose a barrier to widespread adoption. Tap Systems Inc. reported that based on over 5,000 demos, users could learn how to tap 9 letters within an average of 3 minutes. Moreover, most users could achieve 35 WPM within 21 days by practicing for just 10 minutes per day [65]. This demonstrates the potential of the tapping-based input method. In the future, incorporating two wristband devices could facilitate a typing experience closer to the QWERTY layout of a physical keyboard, significantly reducing the learning curve [82].

9 CONCLUSION

We present *FineType*, a text entry system that enables near-complete mapping of an entire physical keyboard (including letters, symbols, and numbers) using various finger combinations and three distinct postures. *FineType* uses a wristband equipped with an IMU and an infrared camera positioned beneath the wrist to enable text entry on any flat surface. We collected data for 10 common finger combinations and 3 finger postures and trained a multi-task, multi-label network to detect these gestures. Cross-user validation demonstrated accuracies of 98.26%, 95.53%, and 94.19% for the 10 finger combinations, 3 postures, and all 30 gesture categories, respectively. Additionally, our model achieved prediction accuracies of 91.27% and 93.86% for 8 previously unseen finger combinations and their corresponding postures. By employing a user-adaptive few-shot learning approach, we improved the average accuracy for these 8 unknown combinations to 97.05% across different users. These

results showcase *FineType*'s potential to recognize all possible finger combinations and three postures (i.e. 31 × 3 = 93). Our user study (N=10) showed that participants achieved an average typing speed of 35.1 WPM with a CER of 5.1%, reaching 93% of the speed of single-hand touchscreen typing. Additionally, we further compared the typing efficiency of numbers, symbols and letters with TapXR. Using *FineType*, participants generally completed the study with a lower physical burden and at a faster speed. Users generally found our keyboard mapping method simpler and preferred using our system.

ACKNOWLEDGMENTS

We sincerely appreciate the valuable comments and suggestions from the anonymous reviewers, which have significantly improved the quality of this paper. We also extend our gratitude to all participants who engaged in our user study or contributed data, as their involvement played a crucial role in our research. This work was supported in part by the National Natural Science Foundation of China under Grants 62376132 and 62321005.

REFERENCES

[1] [n.d.]. NASA task load Index (NASA-TLX). https://humansystems.arc.nasa.gov/groups/tlx/downloads/TLXScale.pdf.

[2] Jiban Adhikary and Keith Vertanen. 2021. Typing on Midair Virtual Keyboards: Exploring Visual Designs and Interaction Styles. In *Human-Computer Interaction–INTERACT 2021: 18th IFIP TC 13 International Conference, Bari, Italy, August 30–September 3, 2021, Proceedings, Part IV 18*. Springer, 132–151.

[3] Tomoko Aoki, Peter R Francis, and Hiroshi Kinoshita. 2003. Differences in the abilities of individual fingers during the performance of fast, repetitive tapping movements. *Experimental Brain Research* 152 (2003), 270–280.

[4] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2009. Analysis of text entry performance metrics. In *2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH)*. IEEE, 100–105.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828.

[6] Wenqiang Chen, Lin Chen, Meiyi Ma, Farshid Salemi Parizi, Shwetak Patel, and John Stankovic. 2021. ViFin: Harness Passive Vibration to Continuous Micro Finger Writing with a Commodity Smartwatch. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–25.

[7] Wenqiang Chen, Ziqi Wang, Pengrui Quan, Zhencan Peng, Shupei Lin, Mani Srivastava, Wojciech Matusik, and John Stankovic. 2023. Robust Finger Interactions with COTS Smartwatches via Unsupervised Siamese Adaptation. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–14.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 248–255.

[9] Vivek Dhakal, Anna Maria Feit, Per Ola Kristensson, and Antti Oulasvirta. 2018. Observations on Typing from 136 Million Keystrokes. In *Proceedings of the 2018 CHI conference on Human Factors in Computing Systems*. 1–12.

[10] Yuqiang Ding, Qian Yang, Yannanqi Li, Zhiyong Yang, Zhengyang Wang, Haowen Liang, and Shin-Tson Wu. 2023. Waveguide-based augmented reality displays: perspectives and challenges. *eLight* 3, 1 (2023), 24.

[11] John Dudley, Hrvoje Benko, Daniel Wigdor, and Per Ola Kristensson. 2019. Performance Envelopes of Virtual Keyboard Text Input Strategies in Virtual Reality. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 289–300.

[12] John J Dudley, Keith Vertanen, and Per Ola Kristensson. 2018. Fast and Precise Touch-Based Text Entry for Head-Mounted Augmented Reality with Variable Occlusion. *ACM Transactions on Computer-Human Interaction (TOCHI)* 25, 6 (2018), 1–40.

[13] John J Dudley, Jingyao Zheng, Aakar Gupta, Hrvoje Benko, Matt Longest, Robert Wang, and Per Ola Kristensson. 2023. Evaluating the Performance of Hand-Based Probabilistic Text Input Methods on a Mid-Air Virtual Qwerty Keyboard. *IEEE Transactions on Visualization and Computer Graphics* (2023).

[14] Aarthi Easwara Moorthy and Kim-Phuong L Vu. 2015. Privacy Concerns for Use of Voice Activated Personal Assistant in the Public Space. *International Journal of Human-Computer Interaction* 31, 4 (2015), 307–335.

[15] Anna Maria Feit, Daryl Weir, and Antti Oulasvirta. 2016. How we type: Movement strategies and performance in everyday typing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 4262–4273.

[16] Leah Findlater, Jacob O Wobbrock, and Daniel Wigdor. 2011. Typing on flat glass: examining ten-finger expert typing patterns on touch surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2453–2462.

[17] Xingyu Fu and Mingze Xi. 2024. Typing on Any Surface: Real-Time Keystroke Detection in Augmented Reality. In *2024 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*. IEEE, 350–354.

[18] Hyunjae Gil and Ian Oakley. 2023. ThumbAir: In-Air Typing for Head Mounted Displays. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–30.

[19] Jun Gong, Zheer Xu, Qifan Guo, Teddy Seyed, Xiang'Anthony' Chen, Xiaojun Bi, and Xing-Dong Yang. 2018. WristText: One-handed Text Entry on Smartwatch using Wrist Gestures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–14.

[20] Patrick Grady, Jeremy A Collins, Chengcheng Tang, Christopher D Twigg, Kunal Aneja, James Hays, and Charles C Kemp. 2024. PressureVision++: Estimating Fingertip Pressure from Diverse RGB Images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 8698–8708.

[21] Simon Greenwold. 2003. Spatial computing. *Massachusetts Institute of Technology, Master* (2003).

[22] Yizheng Gu, Chun Yu, Zhipeng Li, Weiqi Li, Shuchang Xu, Xiaoying Wei, and Yuanchun Shi. 2019. Accurate and Low-Latency Sensing of Touch Contact on Any Surface with Finger-Worn IMU Sensor. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 1059–1070.

[23] Yizheng Gu, Chun Yu, Zhipeng Li, Zhaoheng Li, Xiaoying Wei, and Yuanchun Shi. 2020. QwertyRing: Text Entry on Physical Surfaces Using a Ring. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 4 (2020), 1–29.

[24] Charlotte Häger-Ross and Marc H Schieber. 2000. Quantifying the independence of human finger movements: comparisons of digits, hands, and movement frequencies. *Journal of Neuroscience* 20, 22 (2000), 8542–8550.

[25] Ke He, Chentao Li, Yongjie Duan, Jianjiang Feng, and Jie Zhou. 2024. TrackPose: Towards Stable and User Adaptive Finger Pose Estimation on Capacitive Touchscreens. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 4 (2024), 1–22.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

[27] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. 2014. Consumed endurance: a metric to quantify arm fatigue of mid-air interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1063–1072.

[28] Jonggi Hong, Seongkook Heo, Poika Isokoski, and Geehyuk Lee. 2015. SplitBoard: A Simple Split Soft Keyboard for Wristwatch-sized Touch Screens. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 1233–1236.

[29] Qibin Hou, Daquan Zhou, and Jiashi Feng. 2021. Coordinate Attention for Efficient Mobile Network Design. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 13713–13722.

[30] Yi-Ta Hsieh, Antti Jylhä, Valeria Orso, Luciano Gamberini, and Giulio Jacucci. 2016. Designing a Willing-to-Use-in-Public Hand Gestural Interaction Technique for Smart Glasses. In *Proceedings of the 2016 CHI conference on Human Factors in Computing Systems*. 4203–4215.

[31] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2020. CodeSearchNet Challenge: Evaluating the State of Semantic Code Search. arXiv:1909.09436 [cs.LG] https://arxiv.org/abs/1909.09436

[32] Kaori Ikematsu and Shota Yamanaka. 2020. ScraTouch: Extending Interaction Technique Using Fingernail on Unmodified Capacitive Touch Surfaces. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (2020), 1–19.

[33] Tap Systems Inc. 2021. *Tap Strap 2*. https://www.tapwithus.com/product/tap-strap-2/

[34] Tap Systems Inc. 2023. *Introducing TapXR*. https://www.tapwithus.com

[35] Sujin Jang, Wolfgang Stuerzlinger, Satyajit Ambike, and Karthik Ramani. 2017. Modeling Cumulative Arm Fatigue in Mid-Air Interaction based on Perceived Exertion and Kinetics of Arm Motion. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 3328–3339.

[36] David Kim, Otmar Hilliges, Shahram Izadi, Alex D Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 2012. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. 167–176.

[37] Taejun Kim, Amy Karlson, Aakar Gupta, Tovi Grossman, Jason Wu, Parastoo Abtahi, Christopher Collins, Michael Glueck, and Hemant Bhaskar Surale. 2023. STAR: Smartphone-analogous Typing in Augmented Reality. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–13.

[38] Bryan Klimt and Yiming Yang. 2004. The Enron Corpus: A New Dataset for Email Classification Research. In *European Conference on Machine Learning*. Springer, 217–226.

[39] Marion Koelle, Matthias Kranz, and Andreas Möller. 2015. Don't look at me that way! Understanding user attitudes towards data glasses usage. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Service*. 362–372.

[40] Falko Kuester, Michelle Chen, Mark E Phair, and Carsten Mehring. 2005. Towards keyboard independent touch typing in VR. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. 86–95.

[41] Chentao Li, Jinyang Yu, Ke He, Jianjiang Feng, and Jie Zhou. 2024. SwivelTouch: Boosting Touchscreen Input with 3D Finger Rotation Gesture. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 2 (2024), 1–30.

[42] Chao Lian, Xianshou Ren, Yuliang Zhao, Xueliang Zhang, Ruoyu Chen, Shuyu Wang, Xiaopeng Sha, and Wen J Li. 2020. Towards a Virtual Keyboard Scheme Based on Wearing One Motion Sensor Ring on Each Hand. *IEEE Sensors Journal* 21, 3 (2020), 3379–3387.

[43] Chen Liang, Xutong Wang, Zisu Li, Chi Hsia, Mingming Fan, Chun Yu, and Yuanchun Shi. 2023. ShadowTouch: Enabling Free-Form Touch-Based Hand-to-Surface Interaction with Wrist-Mounted Illuminant by Shadow Projection. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–14.

[44] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*. 2980–2988.

[45] Zongjian Liu, Jieling He, Jianjiang Feng, and Jie Zhou. 2023. PrinType: Text Entry via Fingerprint Recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–31.

[46] Janeen D Loehr and Caroline Palmer. 2007. Cognitive and biomechanical influences in pianists' finger tapping. *Experimental brain research* 178 (2007), 518–528.

[47] I Scott MacKenzie and R William Soukoreff. 2003. Phrase sets for evaluating text entry techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*. 754–755.

[48] Ben Maman and Amit Bermano. 2022. TypeNet: Towards Camera Enabled Touch Typing on Flat Surfaces through Self-Refinement. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 1140–1149.

[49] Anders Markussen, Mikkel R Jakobsen, and Kasper Hornbæk. 2013. Selection-Based Mid-Air Text Entry on Large Displays. In *Human-Computer Interaction–INTERACT 2013: 14th IFIP TC 13 International Conference, Cape Town, South Africa, September 2-6, 2013, Proceedings, Part I 14*. Springer, 401–418.

[50] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. 2014. Vulture: a mid-air word-gesture keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1073–1082.

[51] Sven Mayer, Huy Viet Le, and Niels Henze. 2017. Estimating the Finger Orientation on Capacitive Touchscreens Using Convolutional Neural Networks. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. 220–229.

[52] Manuel Meier, Paul Streli, Andreas Fender, and Christian Holz. 2021. TapID: Rapid Touch Interaction in Virtual Reality using Wearable Sensing. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. IEEE, 519–528.

[53] Takehiro Niikura, Yoshihiro Watanabe, and Masatoshi Ishikawa. 2014. Anywhere surface touch: utilizing any surface as an input area. In *Proceedings of the 5th Augmented Human International Conference*. 1–8.

[54] Ju Young Oh, Ji-Hyung Park, and Jung-Min Park. 2020. FingerTouch: Touch Interaction Using a Fingernail-Mounted Sensor on a Head-Mounted Display for Augmented Reality. *IEEE Access* 8 (2020), 101192–101208.

[55] Manuel Prätorius, Dimitar Valkov, Ulrich Burgbacher, and Klaus Hinrichs. 2014. DigiTap: an eyes-free VR/AR symbolic input device. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*. 9–18.

[56] Mark Richardson, Fadi Botros, Yangyang Shi, Pinhao Guo, Bradford J Snow, Linguang Zhang, Jingming Dong, Keith Vertanen, Shugao Ma, and Robert Wang. 2024. StegoType: Surface Typing from Egocentric Cameras. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–14.

[57] Mark Richardson, Matt Durasoff, and Robert Wang. 2020. Decoding Surface Touch Typing from Hand-Tracking. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 686–696.

[58] David B Schick and Liron Ilouz. 2023. Wearable finger tap detection system with low power mode. US Patent 11,797,086.

[59] Weinan Shi, Chun Yu, Shuyi Fan, Feng Wang, Tong Wang, Xin Yi, Xiaojun Bi, and Yuanchun Shi. 2019. VIPBoard: Improving Screen-Reader Keyboard for Visually Impaired People with Character-Level Auto Correction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[60] Weinan Shi, Chun Yu, Xin Yi, Zhen Li, and Yuanchun Shi. 2018. TOAST: Ten-Finger Eyes-Free Typing on Touchable Surfaces. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–23.

[61] R William Soukoreff and I Scott MacKenzie. 2003. Metrics for text entry research: An evaluation of MSD and KSPC, and a new unified error metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 113–120.

[62] Paul Streli, Jiaxi Jiang, Andreas Rene Fender, Manuel Meier, Hugo Romat, and Christian Holz. 2022. TapType: Ten-finger text entry on everyday surfaces via Bayesian inference. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–16.

[63] Paul Streli, Mark Richardson, Fadi Botros, Shugao Ma, Robert Wang, and Christian Holz. 2024. TouchInsight: Uncertainty-aware Rapid Touch and Text Input for Mixed Reality from Egocentric Vision. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–16.

[64] Ryo Takahashi, Masaaki Fukumoto, Changyo Han, Takuya Sasatani, Yoshiaki Narusue, and Yoshihiro Kawahara. 2020. TelemetRing: A Batteryless and Wireless Ring-shaped Keyboard using Passive Inductive Telemetry. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 1161–1168.

[65] Inc. Tap Systems. 2025. *Invest in Tap Systems, Inc.: Command AI Powered Devices With A Pinch Tap Or Swipe.* https://wefunder.com/tapwithus/ Backup available at https://archive.ph/YcPZu.

[66] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008).

[67] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. 2021. Multi-Task Learning for Dense Prediction Tasks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 7 (2021), 3614–3633.

[68] Keith Vertanen and Per Ola Kristensson. 2011. A versatile dataset for text entry evaluations based on genuine mobile emails. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. 295–298.

[69] 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*. 659–668.

[70] Jonas Vogelsang, Francisco Kiss, and Sven Mayer. 2021. A Design Space for User Interface Elements using Finger Orientation Input. In *Proceedings of Mensch und Computer 2021*. 1–10.

[71] Cheng-Yao Wang, Wei-Chen Chu, Po-Tsung Chiu, Min-Chieh Hsiu, Yih-Harn Chiang, and Mike Y Chen. 2015. PalmType: Using Palms as Keyboards for Smart Glasses. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 153–160.

[72] Daryl Weir, Henning Pohl, Simon Rogers, Keith Vertanen, and Per Ola Kristensson. 2014. Uncertain text entry on mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2307–2316.

[73] Eric Whitmire, Mohit Jain, Divye Jain, Greg Nelson, Ravi Karkar, Shwetak Patel, and Mayank Goel. 2017. DigiTouch: Reconfigurable Thumb-to-Finger Input and Text Entry on Head-mounted Displays. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 1–21.

[74] Wikipedia. 2024. Letter frequency. https://en.wikipedia.org/wiki/Letter_frequency

[75] Robert Xiao, Julia Schwarz, and Chris Harrison. 2015. Estimating 3d finger angle on commodity touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*. 47–50.

[76] Robert Xiao, Julia Schwarz, Nick Throm, Andrew D Wilson, and Hrvoje Benko. 2018. MRTouch: Adding Touch Input to Head-Mounted Mixed Reality. *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (2018), 1653–1660.

[77] Xin Yi, Chun Yu, Mingrui Zhang, Sida Gao, Ke Sun, and Yuanchun Shi. 2015. ATK: Enabling Ten-Finger Freehand Typing in Air Based on 3D Hand Tracking Data. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 539–548.

[78] Chun Yu, Yizheng Gu, Zhican Yang, Xin Yi, Hengliang Luo, and Yuanchun Shi. 2017. Tap, Dwell or Gesture? Exploring Head-Based Text Entry Techniques for HMDs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 4479–4488.

[79] Shumin Zhai, Per-Ola Kristensson, and Barton A Smith. 2005. In search of effective text input interfaces for off the desktop computing. *Interacting with Computers* 17, 3 (2005), 229–250.

[80] Mingrui Ray Zhang and Jacob O Wobbrock. 2019. Beyond the Input Stream: Making Text Entry Evaluations More Flexible with Transcription Sequences. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 831–842.

[81] Mingrui Ray Zhang, Shumin Zhai, and Jacob O Wobbrock. 2019. Text Entry Throughput: Towards Unifying Speed and Accuracy in a Single Performance Metric. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[82] Mingrui Ray Zhang, Shumin Zhai, and Jacob O Wobbrock. 2022. TypeAnywhere: A QWERTY-Based Text Entry Solution for Ubiquitous Computing. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–16.

[83] Jingjie Zheng, Blaine Lewis, Jeff Avery, and Daniel Vogel. 2018. FingerArc and FingerChord: Supporting Novice to Expert Transitions with Guided Finger-Aware Shortcuts. In *Proceedings of the 31st Annual ACM Symposium on User Interface*

Software and Technology. 347–363.
[84] Suwen Zhu, Tianyao Luo, Xiaojun Bi, and Shumin Zhai. 2018. Typing on an Invisible Keyboard. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

A USER-ADAPTIVE NEW GESTURE RECOGNITION

In Section 5.2, we verified that our model achieved an accuracy of 85.32% on the new gesture set, where the recognition rate is primarily limited by the inaccurate recognition of finger combinations. We expect to achieve higher recognition accuracy for users with new gestures, ideally without needing to collect excessive data and by simply performing a few registrations, thus enhancing the user experience. We will now introduce our user-adaptive gesture recognition framework.

A.1 Model Architecture

In Section 5.1, we validated that our model performs well on the original gesture set. We aim to transfer the model's feature extraction capabilities to a new set of gestures and design an additional lightweight prediction head for classifying new gestures. When users introduce new gestures, the finger postures are predefined categories, and only the new finger combinations are the newly defined elements that primarily affect the model's recognition performance. Therefore, our model primarily uses transfer learning and few-shot learning methods to enhance recognition performance for new finger combinations. We freeze the feature extraction layers trained on the original gesture set and add a new classification head for each new user, as shown in Fig. 14. Our approach customizes an additional classification head for each user to classify new finger combinations, with 21 possible new combinations available (i.e., 2^5 - 10 predefined gestures - 1 non-tap). We use a lightweight two-layer fully connected network to predict the new finger combinations added by users. The first layer is a linear layer with 32 nodes, equipped with a ReLU activation function and a dropout layer ($p = 0.8$) to reduce overfitting; the second layer is a linear layer with N+1 nodes (where N represents the number of new gestures the user needs to add, plus 1 for previously defined gestures) connected to a softmax function to predict the probability of each category. Since our model freezes the weights of the feature extractor and only trains a lightweight fully connected prediction head, it does not require backpropagation across all model parameters, uses minimal VRAM, and the training process is very quick.

We used the 2D t-SNE [66] method to plot the embeddings of 11 old finger tapping combinations collected in Section 3.1 and 8 new tapping combinations from two new users collected in Section 5.2.1 after feature extraction. Observation showed that the old 11 gestures formed distinct clusters, demonstrating the performance of the model's feature extractor and the separability of the original gestures in the hidden space. Additionally, each of the newly defined 8 gestures also formed clear clusters with distinct decision boundaries from other gesture categories. These findings support our approach of customizing a new prediction head for tapping fingers for each new user.

Note that in practical applications, the existing model's prediction branches work as usual. If the new network prediction branch identifies an old gesture, it uses the results from the existing finger combination prediction branch; if it predicts a new gesture, it uses the results from the new gesture prediction. Throughout this process, the branch predicting finger postures remains unchanged.

A.2 Model Training

A.2.1 Dataset Splitting. For each user, we added 8 gestures from Section 5.2.2 as new finger combinations, making the fully connected network's final output 9 (i.e., 8 new gestures plus 1 category for old gestures). For a new user, there are 8 new finger combinations, with 18 gestures collected for each combination (6 each for Pose 1, Pose 2, and Pose 3). We split the data for each pose into training, validation, and test sets in a 2:1:3 ratio. Thus, each user has 48 training samples, 24 validation samples, and 72 test samples. A 3-shot setup means each pose for every finger combination appears only once, resulting in $1 \times 3 \times 8 = 24$ training samples. In contrast, a 6-shot setup means each pose appears twice, using all $2 \times 3 \times 8 = 48$ training samples. For the old gesture data, we divided the data collected in Section 4.2 into training, validation, and test sets in a 4:1:5 ratio.

A.2.2 Few-shot Learning. We used a few-shot learning method, training the model with a small amount of new registration data from users. Due to the scarcity of training data for new gestures, the model is likely to overfit. To address this, we employed data augmentation techniques. We used the five data augmentation methods from Section 4.4.1, randomly combining them in $2^5 - 1$ different ways to significantly expand the data pool for new diversity gesture samples. Additionally, to better focus on features of categories with fewer samples, we adopted the Focal loss [44] function during training to enhance the model's recognition performance for new gestures, setting parameters $\gamma = 2$ and $\alpha = 0.9$.

A.3 Model Performance

For each user, we selected the model with the highest F1 score on the validation set for testing. We evaluated the performance of models using the original prediction, 3-shot, and 6-shot methods, as shown in Fig. 16a. It can be observed that customizing the model to adapt to each user improves the recognition of new gestures to some extent. Moreover, as we increase the number of gestures registered by the user, the model's performance significantly improves, with accuracy increasing from 92.71% (3-shots) to 97.05% (6-shots). Fig. 16b displays the confusion matrix for the overall classification of the test set samples using the 6-shot learning model. This proves that our model, while recognizing new gestures, also maintains compatibility with old gestures.

A.4 Method Comparison

We compared two approaches using the same training, validation, and test datasets from Section A.2.1: fine-tuning only the finger combination classification components of the network (Ft-Part) and fine-tuning the entire network (Ft-Full). Both methods yielded similar test accuracy results, as summarized in Table 5. We also evaluated the training speed by conducting a single epoch of training on a dataset containing 3,000 samples with a batch size of 300, using an NVIDIA GeForce RTX 3090 GPU. Our method, which requires no annotations, achieves training speeds two orders of

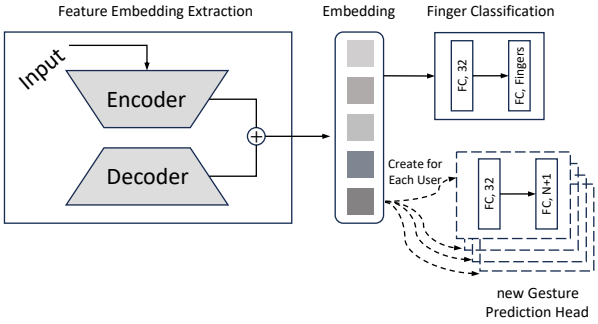


Figure 14: User-adaptive new gesture network structure diagram. We extract the embeddings before the finger combination classifier in the original network structure and create a new classifier for new gesture recognition.

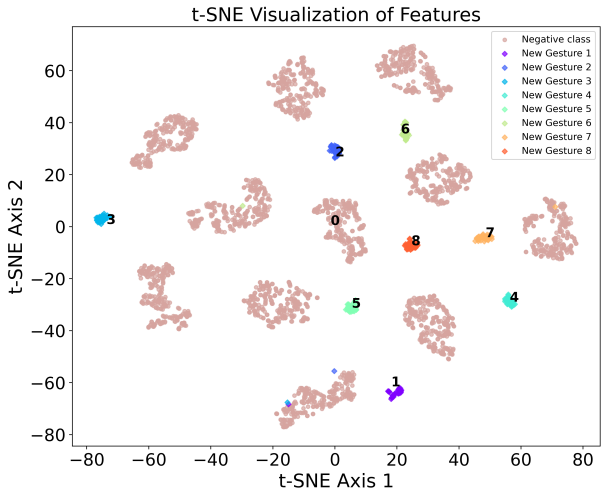
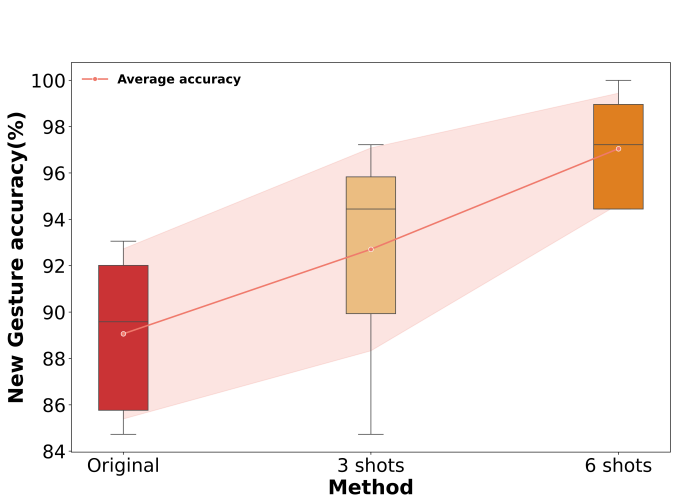
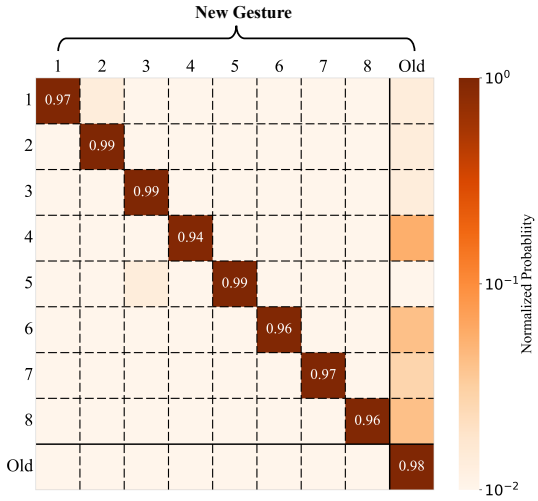


Figure 15: 2-D t-SNE visualization of feature embeddings corresponding to different finger combination gestures. Label 0 represents the 11 old finger combination gestures. Labels 1-8 correspond to the new 8 gesture categories. The new gesture samples are from two users, while the old gesture data is from the original training dataset. The feature extractor is trained on the old gesture set.



(a)



(b)

Figure 16: Performance of the User-Adaptive Model: (a) box plot of average accuracy for recognizing new finger combinations under different shots. The red dot represents the average accuracy, and the pink shaded curve indicates the standard deviation. (b) 8 new gesture recognition confusion matrix.

magnitude faster and involves three orders of magnitude fewer trainable parameters compared to other fine-tuning-based methods. Despite these advantages, its accuracy closely approaches that of fully fine-tuning the network with manually annotated fingertip heatmaps.

Table 5: Comparison of our method with Ft-Part and FT-Full in terms of fingertip annotation , Training Time (s), Trainable Parameters (M), and test Accuracy (%).

Method	w/o fingertip annotation	Training time	Training Params	Accuracy
Ours	✔	0.015	0.03	97.05
Ft-Part	✔	7.96	28.72	96.18
Ft-Full	✗	11.23	29.19	97.4

Table 6: Task 2 phrase set: percentage (%) of letters, numbers, and symbols across different blocks.

Block	Letter	Digit	Symbol
1	49.3	17.9	32.8
2	33.3	25.4	41.3
3	40.4	9.6	50

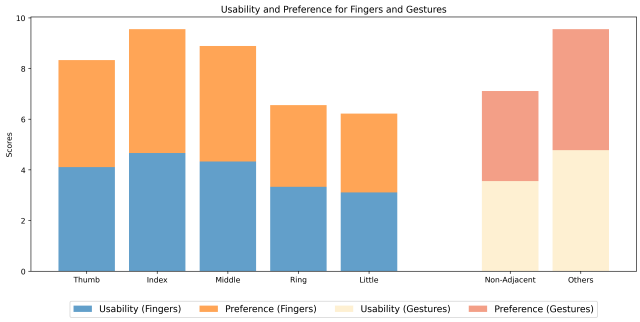


Figure 17: A stacked bar chart showing usability and preference ratings (5-point Likert scale) across fingers and gestures.

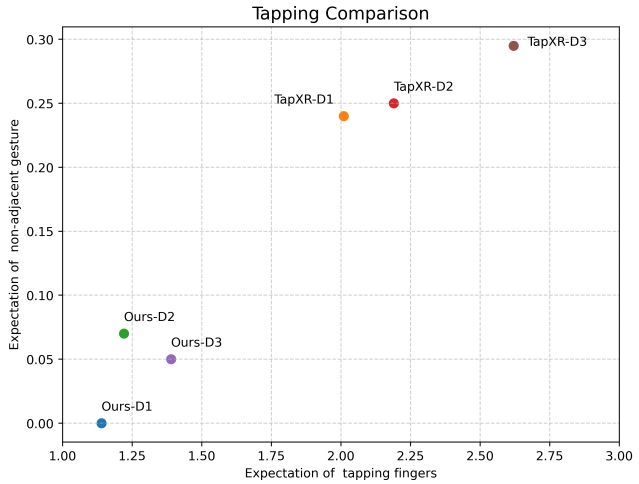


Figure 18: A comparison between *FineType* and TapXR at the tapping level: the x-axis represents the average number of fingers used per tap, and the y-axis represents the average number of non-contiguous gestures. D1, D2, and D3 correspond to the MacKenzie and Soukoreff's phrase set, the Enron Email Dataset, and CodeSearchNet, respectively.

Table 7: The average accuracy (%) for various ablation experiments and our original method across 8 new finger combinations, 3 finger postures, and all 24 categories. (Here, '•' indicates the finger is pressed down, and '○' indicates the finger is lifted. For example, "• • • ○ ○" means the thumb, index finger, and middle finger are pressed, while the ring finger and little finger are not pressed.)

Method	Finger Combination								Posture	Overall
	•○•○○	•○○•○	•○○○•	○•○•○	○•○○•	•••○○	••••○	•••••		
w/o Augmentation	80.16	96.03	96.83	52.39	88.89	82.54	79.37	84.92	86.61	70.93
w/o Auxiliary Task	0	19.84	16.67	3.17	77.78	0	1.59	0	91.27	12.30
Ours	86.51	97.62	98.41	90.48	92.86	88.10	83.33	92.86	93.86	85.32