Unikey: Enabling Surface-Based Typing with Commodity Smartwatches via Cross-Modal Learning

Sean Rui Xiang Tan
National University of Singapore
strx@nus.edu.sg

Mun Choon Chan
National University of Singapore
chanmc@comp.nus.edu.sg

Jun Han
KAIST
junhan@cyphy.kaist.ac.kr

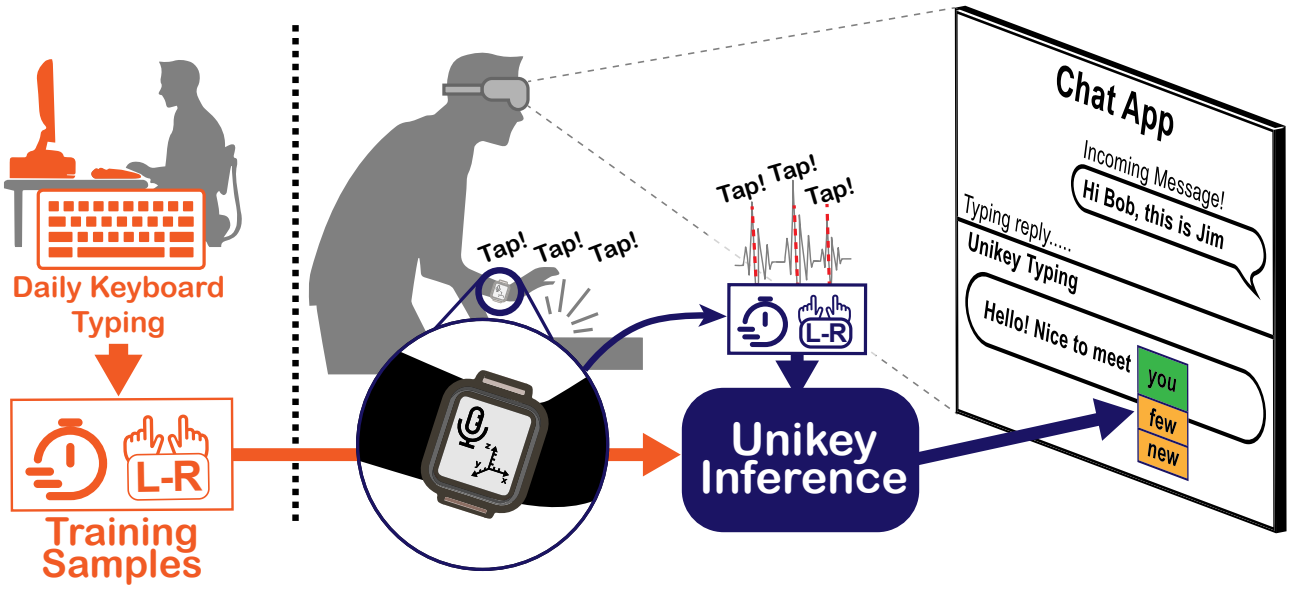


Figure 1: Figure illustrates our vision for how *Unikey* can enable users to type on any surface without a physical keyboard, using only a accelerometer and microphone aboard a commodity smartwatch to sense keystrokes. What sets *Unikey* apart is its out-of-the-box functionality, requiring no lengthy collection of training data on watch sensors. Instead, it applies typing patterns learned from a user's regular typing on physical keyboard to infer typed text.

Abstract

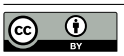
With the rise of mobile technologies such as augmented and virtual reality (AR/VR) and wearables, as well as smart TVs, the large form factor of traditional keyboards is becoming increasingly impractical. Sensing typing on surfaces offers a potential alternative, but most current solutions rely on either expensive, custom hardware or extensive user bootstrapping and calibration. In this paper, we envision *Unikey*, an approach to surface-based typing that uses only a commodity smartwatch to detect finger taps on a flat surface. By seamlessly adapting to the user's existing typing habits on conventional keyboards, *Unikey* eliminates the need for both specialized equipment and burdensome sensor data collection, reducing overhead for users. To demonstrate the feasibility of our approach, we implement a proof-of-concept and evaluate our technique with comprehensive real-world experiments under varying conditions. Participants were invited to type while wearing smartwatches, resulting in over 2,700 minutes of recorded typing. Our experiments show that *Unikey* can achieve an equivalent average top-5 word error rate of 6.45%, indicating a potential solution simple, everyday text-entry tasks.

CCS Concepts

• **Human-centered computing** → **Ubiquitous and mobile devices**; **Text input**; **Mixed / augmented reality**; **Keyboards**.

Keywords

Text entry, ubiquitous computing, virtual reality, smartwatch, keystroke dynamics



UIST '25, September 28-October 1, 2025, Busan, Republic of Korea

ACM ISBN 979-8-4007-2037-6/2025/09
https://doi.org/10.1145/3746059.3747611

ACM Reference Format:
Sean Rui Xiang Tan, Mun Choon Chan, and Jun Han. 2025. *Unikey*: Enabling Surface-Based Typing with Commodity Smartwatches via Cross-Modal Learning. In *The 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25), September 28-October 1, 2025, Busan, Republic of Korea.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3746059.3747611

1 Introduction

The rise of mobile technologies such as augmented/virtual reality (AR/VR) and wearable devices (i.e., smartwatches, smart rings), along with the prevalence of smart TVs, has rendered the traditional keyboard increasingly impractical [2, 64]. These modern devices offer a seamless and integrated experience with technology, but demand input interfaces that align with their form factor or mobile nature, which conventional keyboards are not well-suited. For example, AR smartglasses (i.e., Optical Head Mounted Displays) like the Xreal Air, Rayneo Air2, and EvenRealities G1 overlay digital displays unto a user's vision but lack typing interfaces, disrupting the user experience [8, 10, 11]. Similarly, a user wearing a VR headset like the Oculus Quest struggles to type with a traditional keyboard, as they cannot see the keys, making accurate typing difficult without breaking immersion [3, 7, 54]. While some devices do have in-built text entry, typing with such interfaces is often still limited by form factor or design. For example, typing using a smartwatch's tiny on-screen keyboard is challenging, since the button for each character is often smaller than a fingertip [117]. Using a smart TV remote control for typing is equally tedious, as navigating an on-screen keyboard with arrow keys is slow and frustrating. Moreover, entering passwords on such devices raises privacy concerns, as sensitive information may be visible to others also watching the screen [1, 9, 29, 59].

While alternative solutions like voice commands and gesture-based typing exist to address the limitations of traditional keyboards, each comes with its own challenges. Voice Commands raise usability and privacy concerns in speech-unfriendly environments [71, 121], and gesture-based systems often require a steep learning curve, making them less intuitive [27, 34, 36, 43, 63, 70, 101, 113, 118].

A promising approach is the surface-based text entry, which recreates the keyboard-like typing experience on surfaces, without the need for actual physical keyboard hardware, making it ideal for integration into AR/VR, wearable devices, and smart TVs. While effective, many existing techniques for surface-based typing are limited by dependence on costly customized hardware (i.e., gloves, wristbands, armbands), which create deployability and usability challenges and hinder widespread adoption [15, 53, 102, 120, 123]. Others use ubiquitous commercial-off-the-shelf (COTS) devices (i.e., smartphones), but require significant setup each time the user types [109, 120]. Critically, we observe that these limitations stem from reliance on *precise sensor-level features* (e.g., fine-grained fingerprints in the acoustic spectrum) to disambiguate between different typed characters or words, which additionally burdens users with the deliberate collection of sensor data for training, which can be time-consuming and inconvenient, and prevents a pre-configured, out-of-the-box experience [46, 73, 109, 115].

In light of these limitations, we explore a novel paradigm for surface-based text-inference that leverages patterns in typing behavior. We rely on two key observations: (1) these patterns, like specific finger movements or keystroke timing of key sequences (i.e., words), develop as habits over time as users become accustomed the layout of the physical keyboard. (2) Since typing on surfaces uses an identical layout, typing behavior patterns in both modalities are highly correlated. Therefore, learning the relationship between patterns to typed-words on a physical keyboard also enables us to apply these patterns to infer text that is typed on surfaces. Since these patterns can be learned as a byproduct of observing users typing in text processors like Google Docs or Gmail, our approach avoids the need for high-fidelity sensor training data.

To demonstrate how this approach can be applied in practice, we develop *Unikey*, a proof-of-concept that uses only the microphone and accelerometer on a single COTS smartwatch to enable typing on surfaces. We do not envision *Unikey* as a complete replacement for physical keyboards, but as a system ideal for quick and convenient text-entry (i.e., replying to messages, talking to AI assistants) in mobile, networked, or wearable contexts, where a display is readily available, but text-entry interfaces are limited. Figure 1 illustrates one such usage scenario for *Unikey*. In this example, the user, equipped with a VR headset and engaged in a chat application, types on a tabletop using *Unikey*. There is no requirement for bootstrapping using the smartwatch, as *Unikey* seamlessly integrates the user's keystroke patterns collected from conventional keyboards into the virtual environment.

Unikey's core challenge is transferring typing patterns from conventional physical keyboards to surface-based typing. This transfer is not straightforward because the typical modality of the output of a physical keyboard (i.e., discrete keystrokes) is vastly different from smartwatch sensors (i.e., audio, acceleration), with no obvious and direct comparison.

To address this, we encode typing behavior as a combination of ***coarse-grained temporal and spatial features*** that are shared by both modalities (see § 4.1). *Unikey*'s noise resilient pipeline uses the smartwatch microphone to capture *Inter-Keystroke-Timings* (temporal), while the accelerometer determines whether a key was pressed with the left or right hand (spatial) (see § 4.2). Encoded in this manner, *Unikey* can learn a user's typing behavior on the physical keyboard, and use this knowledge to aid in surface-based text inference (see § 4.3.1).

While this encoding enables *Unikey*'s cross-modal learning, its coarse-grained nature inevitably results in ambiguity in word predictions; words with similar typing patterns may be confused, leading to incorrect outputs. Sources of noise such as (1) natural variability in an individual's typing, and (2) changes in the typing environment, also increase likelihood of accurate predictions. As a result, *Unikey* uses a large language model (LLM) to provide contextual information that guides the inference process (see § 4.3.2). In essence, for each typing attempt, the encoded typing behavior produces a small set of possible words with similar typing patterns, which LLM attempts to disambiguate based on context. Since contextual text prediction alone can also produce ambiguous results, combining *Unikey*'s encoding with an LLM allows *Unikey* to only choose words that match both typing pattern and context, improving the accuracy of text predictions.

We evaluate our prototype by conducting comprehensive experiments using real-world typing data collected from participants typing while wearing smartwatches, and validate our design choices through analysis of system modules. In total, we collect over 2700 minutes of typing and demonstrate that *Unikey* achieves an average *Retry Rate* (i.e., top-5 word error rate) of 6.45%, suggesting that users would rarely need to make corrections, which highlights the

potential of our approach as part of a solution for future practical surface-based typing.

To summarize our contributions:

- We propose a novel approach to surface-based text-entry, leveraging only timing and coarse-grained spatial information for text inference.
- We show empirically that the relationship between typing behavior extracted from everyday keyboard typing can be mapped to typing behavior on surfaces, and these features are sufficient for text inference.
- We design *Unikey*, a bare-bones, proof-of-concept, technical prototype that applies this approach using a COTS smartwatch.
- We evaluate *Unikey* to demonstrate that our approach is able to achieve a reasonable text-entry accuracy while being able to generalize across users, surfaces, and environments.

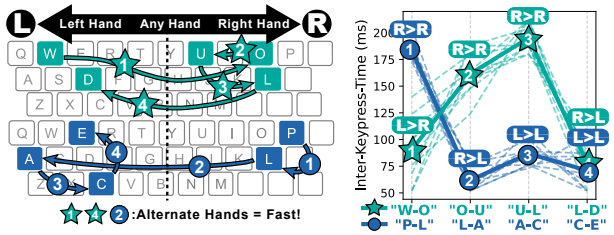


Figure 2: Figure depicts the typing process and the actual average *Inter-Keystroke-Timing*'s for the words "would" and "place" for one of *Unikey*'s participants. For *Inter-Keystroke-Timing*'s, solid lines denote the average for each word, while dotted lines denote individual instances. In this example, the combination of the fingers or hands used and the key position results in very distinct *Inter-Keystroke-Timing*'s for each word.

2 Related Work

We present related research in this domain including prior SoTA Surface-based Typing, other *sensor-based text inference*, and *keystroke timing*.

SoTA Surface-based Typing. Surface-based text-entry is a well studied area of research, with a variety of approaches for accomplishing text inference. We present the most relevant prior works that offer similar surface-based typing using a standard QWERTY layout, and highlight the fundamental limitation of these approaches. First, TypeAnywhere [123] and TapType [102] infer text by determining which finger is used to tap on the keyboard using fine-grained analysis of IMU data, requiring custom wearables on each finger (TypeAnywhere) or custom IMU-enabled wristbands on each hand (TapType), and mapping the finger sequences to characters or words. Meanwhile, Magstroke [15], CamK [119], and UbiK [110] focus on localizing keystrokes using sensors of various modalities on a nearby smartphone. Magstroke uses changing magnetic fields caused by user-worn magnetic rings, CamK uses computer vision with the smartphone camera, and UbiK uses spectral analysis of tap audio. Regardless of sensing modality, these prior works rely heavily on *precise and fine-grained* analysis of raw sensor output, which can be susceptible to a myriad of changing factors such as user typing style, sensor precision, and environmental noise. To account for this, these works similarly use additional hardware (i.e., magnetic rings) or require significant set-up and calibration (i.e., careful alignment of the smartphone to typing area, calibration taps for every key on the keyboard) each time a user types, which decreases the usability of these solutions. *Unikey* stands apart from these prior works by providing a text inference using only a single wearable COTS smartwatch and a short 3-tap calibration sequence on each hand, while only using data that can be collected *incidentally* for bootstrapping.

Sensor-Based Text Inference. Aside from the aforementioned selections, multiple other works [26] propose to detect and infer typed-text (with a QWERTY layout or otherwise) in varied sensor modalities and contexts, which also suffer from similar hardware, bootstrapping, or calibration-related drawbacks. These include custom wearables [83], IR projections [52, 95], depth sensors [42], piezoelectric vibration sensor [73], millimeter-wave radar [58], VR-headsets [53, 94], and even electromyography (EMG) sensors [115]. From an adversarial perspective, many keystroke eavesdropping attacks also attempt to covertly accomplish a similar goal using microphones [22, 31, 44, 60, 66, 74, 85, 99, 108, 111, 124], IMUs [76, 84, 107] and WiFi [33, 56, 105, 114]. While determining the exact fine-grained identity of each keystroke's key is difficult, these prior works indicate the presence or absence of keystroke tap itself can be detected from a variety of different sensors, which inspires our efforts toward a more coarse-grained approach to sensing.

Exploiting Keystroke Timing. Keystroke dynamics is the study of the unique typing patterns of individuals, a large part of which consists of keystroke timing characteristics, which have been shown to be consistent enough to be used for applications like authentication [24, 25, 28, 38, 50, 51, 55, 81, 91, 93] and password hardening [16, 23, 37, 75, 80, 92, 103, 106]. In the adversarial sense, some works have also shown that timing information can be used to disambiguate URLs [61, 72] or guess passwords [100, 122], albeit in an offline and brute force manner.

Unikey's use of keystroke timing is inspired by these works, which seem to indicate that it is both predictable, and contains useful information for text inference. However, unlike an attacker who can generate a large set of possibilities to brute-force and tolerate a low success rate, to provide functional text entry, *Unikey* must regularly infer the correct text within the top few possibilities.

3 Background

We explain the core mechanism behind spatial and temporal patterns used for text inference.

Spatial Location. Keystroke spatial location is inherent to keyboard design and is the most straightforward way to determine typed text. In most common keyboard layouts (i.e., QWERTY, DVORAK, etc. [30]), each key corresponds to a specific character based on its spatial position. At a fine-grained level, accurate spatial positioning allows for easy text inference, since each position corresponds to a unique key. However, as spatial information becomes more coarse (e.g., approximate region of keystroke), the ambiguity in text inference increases as multiple keys may correspond to the

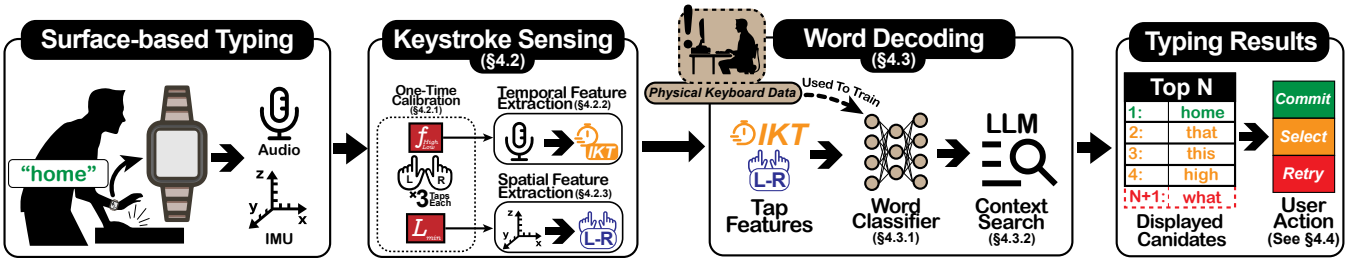


Figure 3: Figure outlines *Unikey*'s pipeline. User taps are captured and processed by the *Keystroke Sensing* module, which extracts coarse-grained tap features. The *Word Decoding* module uses these features to classify the taps into word candidates, and the *Context Searcher*, powered by a Large Language Model (LLM), refines these predictions using the user's previous context. The top N candidates are then displayed, allowing the user to commit, select, or retry until the correct word is entered.

same spatial region. Taken to the extreme, spatial information can be reduced to binary left/right (L-R) hand information, resulting in maximal ambiguity in text inference. While ambiguous, and partially driven by typing style, L-R still encodes some spatial information, as the left hand is largely responsible for typing the left half of the keyboard and vice versa. Given the sensor constraints of a commodity smartwatch, *Unikey* employs coarse-grained L-R information as one component of its text inference pipeline.

Temporal *Inter-Keystroke-Timing*. Temporal timing patterns refer to the relationships between different events (i.e., pressing or releasing keys) when typing. Since only the striking of the surface is detectable by the smartwatch sensors, we focus on *Inter-Keystroke-Timings* (*IKT*), which refers to the time elapsed between consecutive key-presses. In general, trends in *IKTs* have been observed by others (i.e., typing character pairs that are far apart is often faster using alternate hands, while typing the same or closely spaced character pairs is often slower). However, between specific individuals or even words, such trends often break down, since *IKTs* are influenced not only by the fixed position of each key (i.e., keyboard layout), but also the finger used to strike the key, which can vary significantly based on the user's typing style, habits and typing idiosyncrasies [17, 18, 39, 45, 98, 125]. However, for fixed keyboard layout and regularly used words, each user's typing habits and thus *IKTs* exhibit remarkable consistency over time [81]. Figure 2 presents a demonstrative real-world example from one participant in *Unikey*'s experiments. Two observations show the feasibility of using *IKTs* for text inference: (1) the *IKT* for the same word are well clustered, indicating high consistency; and (2) a clear distinction between the *IKTs* of each word, indicating the potential for disambiguation. Since keyboard layout is the same for both physical and surface-based typing, we hypothesize significant similarity between the *IKT* patterns in both scenarios.

Unikey combines limited spatial and temporal properties available to commodity smartwatches. To minimize sensor data collection, *Unikey* reuses data from physical keyboards arising from the user's daily typing events. We discuss the implementation of these strategies in the following section.

4 Design and Implementation

We now present the design and implementation details for *Unikey*. Although not a complete solution for general purpose text-entry, our prototype allows for the testing of our coarse-grained, cross-modal approach on real-world data.

4.1 Design Overview

We illustrate the *Unikey*'s overall pipeline in Figure 3. *Unikey* begins with a brief, three-taps-per-hand calibration for each new surface, letting it adapt microphone and accelerometer data to the environment. As the user types (e.g., "home"), the *Keystroke Sensing* module captures *temporal* (timing) and *spatial* (left/right hand) features. These features are passed to the *Word Classifier*, which is pre-trained on data from the user's ordinary physical keyboard typing. Next, the *Context Searcher* LLM refines the classifier's results based on prior typed context, returning a top-N list of likely matches to the user. The user can then "commit" by accepting the top result, "select" among the next few, or "retry" typing if the correct word does not appear. This process repeats until the text is complete (see § 4.4).

4.2 Keystroke Sensing

This module processes audio and accelerometer data to detect tap events despite low signal-to-noise ratios and varying surface conditions. Specifically, the tap audio signals are easily confused with other percussive noise sources (i.e., doors slamming, various click or tap-like sounds), causing false positives. Meanwhile, tap accelerometer signals vary in magnitude based on typing surfaces, making identifying a universal threshold difficult.

Figure 4 depicts the processing pipeline. First, we calibrate audio and acceleration thresholds for each surface. Next, we identify taps based on the audio and compute inter-keystroke timings and character counts. Finally, we check the accelerometer signal around each detected tap to distinguish left vs. right-hand taps.

4.2.1 One-Time-Calibration. Just once per surface, before a user starts to type, *Unikey* requires three reference taps for each hand on the typing surface, which takes at most two to three seconds to collect. From the raw audio, *Unikey* computes the frequency band of interest, bounded by the frequencies f_{High} and f_{Low}, and the deceleration threshold, L_{min}. Empirically, we observe that legitimate taps on common surfaces have a duller "thud"-like sound with higher energy in the lower frequencies (< 1000 Hz). To identify f_{High} and f_{Low}, we look for the frequency bin below 1000Hz with the highest energy using the Fast Fourier Transform (FFT). From

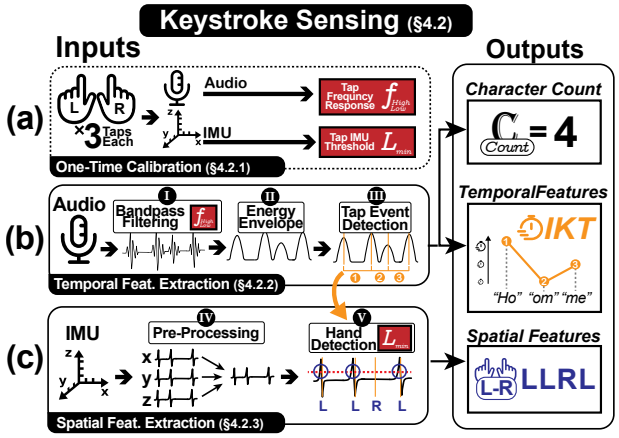


Figure 4: Figure depicts the Keystroke Sensing module. Calibration is only required once, thereafter, the module extracts first *Temporal* followed by *Spatial* features from the user's keystrokes.

the identified peak energy frequency bin, we select the nearest higher (or lower) frequency bin with a 50% drop in energy relative to the peak as f_{High} (or f_{Low}). The user's deceleration threshold, L_{min}, is computed as the mean magnitude of the accelerometer signal in the window of $\pm 10ms$ around each tap event. These calibration parameters, $f_{High/Low}$ and L_{min}, are then used throughout the typing session for audio and accelerometer processing.

4.2.2 Temporal Feature Extraction. This sub-module takes as input the raw audio signal and $f_{High/Low}$ to extract the *IKTs*. By design, the input audio signal is segmented at the word (see § 4.4) level, with guaranteed periods of inactivity between words. The pipeline for audio processing is as follows:

(I) Bandpass Filtering. The raw audio segment is filtered using a fifth-order Butterworth band-pass filter with cut-off frequencies f_{High} and f_{Low}. This emphasizes tap events in the filtered signal while attenuating other sounds in the environment.

(II) Energy Envelope. Next, we extract impulsive high-energy audio in preparation for tap event detection. Using the output of the bandpass filter, we first compute the Root Mean Square (RMS) energy on which we apply a Continuous Wavelet Transform (CWT) with a Morlet wavelet to emphasize transient impulses [41, 97] yielding Aud_{cwt}.

(III) Tap Event Detection. Peaks in Aud_{cwt} correspond to potential tap events, but might still contain false positives. To reduce false positive, we apply peak detection with Constant False Alarm Rate (CFAR) [47, 88] with a sliding window of α frames, setting the dynamic threshold for peak detection as the mean plus β times the standard deviation of each frame of Aud_{cwt}. Empirically, we set α to 51 frames and β to 1.5, which we observe to encapsulate and highlight a single tap from its surrounding noise floor. We also set the minimum time between peaks to 30ms. This value corresponds to the observed lower bound of *IKTs*. The output of this sub-module is the computed *IKTs* plus the corresponding timestamps.

4.2.3 Spatial Feature Extraction. The IMU processing sub-module takes the raw accelerometer signals, x_{raw}, y_{raw}, and z_{raw}, along with the calibration parameter, L_{min}, and the processed audio output to extract the Left-Right (L-R) feature. The IMU processing pipeline is as follows:

(IV) Pre-processing. The raw accelerometer signals, x_{raw}, y_{raw}, and z_{raw} are filtered using a fifth-order Butterworth high-pass filter with a cut-off frequency of 10 Hz to isolate the short-but-impulsive tap events from gravity and other larger-but-slower hand movements. Next, we combine the filtered signals and calculate its L^2 Norm to obtain the overall magnitude of the accelerometer signal.

(V) Hand Detection. To extract the L-R feature, we observe that taps detected in the audio signal correspond to peaks in the accelerometer signal only if the tap was made by the hand wearing the smartwatch. Hence, for each peak extracted from the audio signal, we check if the mean signal strength in the window of $\pm 10ms$ around each peak exceeds the threshold L_{min}. If true, the tap L-R feature corresponds to the hand wearing the smartwatch; otherwise, it corresponds to the tap on the other hand. The L-R features, *IKTs*, are then passed to the Word Decoding module for word prediction.

4.3 Word Decoding

This module infers candidate words from extracted tap features by learning typing patterns from the user's physical keyboard data. While this technique is key to *Unikey*'s use-out-of-box word-inference, it introduces two drawbacks: (1) the user's physical typing patterns might not be perfectly replicated when typing on surfaces, resulting in reduced inference accuracy, and (2) the coarse-grained nature of tap features results in ambiguous inference results.

Figure 5 depicts the word inference pipeline, which addresses these drawbacks. Before deployment, the user's physical keyboard data is augmented to handle differences between keyboard and surface-based typing, which is used to train a classification model. At runtime, the model picks the top K words matching tap features, then uses an LLM to re-rank those candidates based on the user's prior context. Finally, N likely words are shown to the user, who can either accept a match or retry (see § 4.4) if the correct word is missing, keeping track of attempt history to refine future predictions.

4.3.1 Word Classifier. While the overall pattern for *IKTs* remains similar for the same user, we observe that differences in tactile feedback (i.e., from buttons) between physical keys and surface typing can sometimes result in shifts in *IKTs* (see Figure 6(a)). Given that these shifts are not known before deployment, training directly on the user's physical keyboard data can result in severe over-fitting. To mitigate this, we simulate such shifts in physical keyboard *IKTs* during training.

During typing, the *Word Classifier* sub-module takes the extracted *IKTs* and L-R features as input and classifies the top K matching word candidates. We use a small feed-forward multi-layer perceptron with four hidden layers, dropout, and ReLU activations

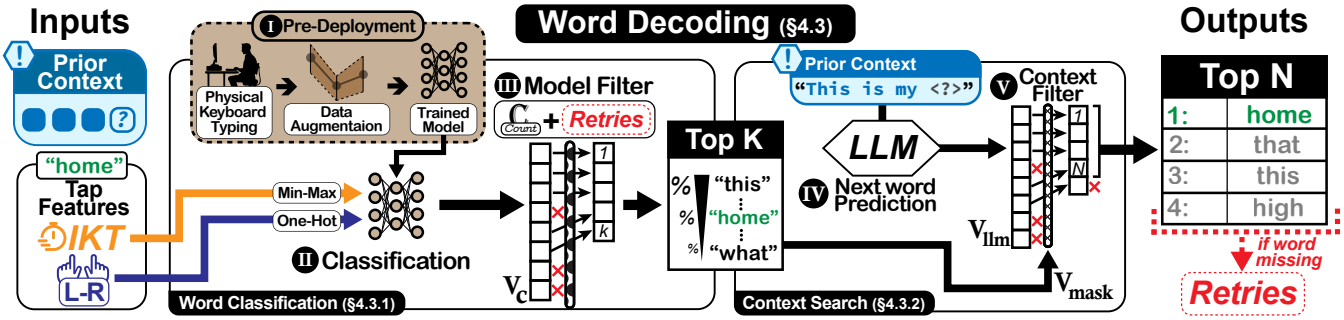


Figure 5: Figure depicts the Word Decoding Module. The module consists of the Word Classification and Context Search sub-modules. The Word Classifier takes as input the *IKTs* and L-R features and outputs the top K most likely word candidates. The Context Searcher uses a Large Language Model (LLM) to re-rank and eliminate prediction candidates with respect to the user's prior typed context, resulting in the final top N candidate words displayed to the user.

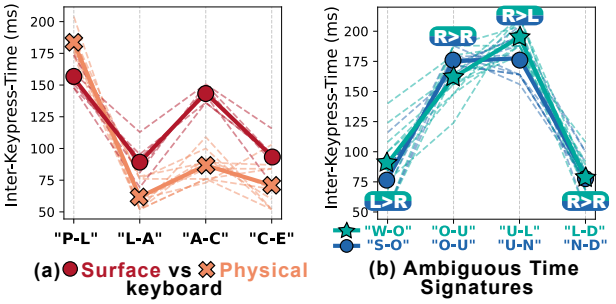


Figure 6: Figure depicts the challenges in inferring typed word from binary L-R and *IKT* features: (a) shifts in *IKTs* between surface and physical keyboard typing, (b) ambiguity in word inference due to similar features for different words.

for our classification model[1]. The model training and classification process is as follows:

(I) Pre-Deployment. To prepare the augmented training set, we first fit individual physical keyboard letter-pair *IKTs* for each word to a skewed-normal distribution [21]. During training, the training *IKTs* are formed by sampling from the corresponding distribution, with an added random offset to simulate the shifts between physical keyboards and surface-based typing. The L-R feature is then reasonably approximated by assigning keys to the left or right hand based on their position on the keyboard, with keys in the middle assigned randomly. The model is trained using supervised learning with the Adam optimizer and the categorical cross-entropy loss function.

(II) Classification. During typing, the input *IKTs* are normalized using min-max scaling, and the binary L-R feature vector is converted to a one-hot encoding. The input vectors are padded to a fixed length of 20 characters. The model classifies these input vectors and outputs a one-hot encoded vector, v_c, representing the likelihood of each word in *Unikey*'s dictionary.

(III) Model filter. Thereafter, we compute the character count by counting the number of taps in the *IKT*. Then, we mask v_c to only consider words with a matching character count, and ignore any previously blacklisted words (see § 4.4). The remaining top K words with the highest likelihoods are then passed to the *Contextual Search* module.

4.3.2 Contextual Search. Recall from § 3 that K (i.e., the output of the *word classifier*) is ambiguous. Similar words can have similar *IKTs* and L-R features, making it difficult to differentiate between them. Since the top K is based primarily on the likelihood of the word given the tap features, the most confident predictions can be often unrelated to the user's prior context. Consider the example in Figure 6(b), in the context of the phrase "That is a very loud [sound]", where "sound" is the target to be predicted. In this case, *Word Classifier* might predict "would" with a higher likelihood than "sound" due to the similarity in features between the two words. However, it is clear from the user's prior context, both semantically and grammatically, that the user is more likely to type "sound" than "would". To account for this, the *Contextual Search* module takes as input the user's prior textual context and the top K words, and uses an LLM to re-rank the top K according to contextual likelihood. We describe the process as follows:

(IV) Next Word Prediction. We use the GPT2 model [90] to predict the next word based on the user's prior context. First, we tokenize the user's context and input it into the LLM. The output logit vector, v_{llm}, represents the likelihood of each word in the GPT2 token dictionary being the next word in the context sequence.

(V) Top K Masking. To map the contextual likelihoods to the top K words, we first tokenize the top K words to form a mask vector v_{mask}. Each element in v_{mask} that corresponds to a word in top K is set to one, and zero otherwise. We then mask v_{llm} with the mass vector v_{mask}, and take the top N words with the highest likelihoods from the masked result as the final output.

4.4 Typing with *Unikey*

We envision a type-and-select model for *Unikey*, akin to standard input method editors (IMEs) for non-Latin alphabet languages (i.e., Chinese, Korean)[4, 6, 32, 96]. After users type a word, and after a configurable delay with no keystrokes, *Unikey* displays the top N word candidates. Figure 7 illustrates the possible user actions: Ⓐ continue typing, automatically selecting the first candidate;

[1]Implementation and hyperparameters can be found at https://github.com/NUS-CIR/UniKey

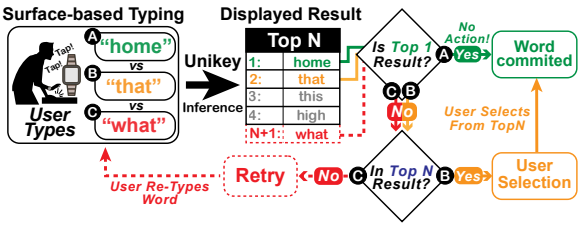


Figure 7: Figure depicts the three cases of user interaction with *Unikey* when typing a word. In the best case where the target word is "home" (A), our system infers the correct word as the top candidate, and the user continues typing without any additional actions. In the second case where the target word is "that" (B), the user selects the word by selecting the correct word from the list of candidates. In the worst case where the target word is "what" (C), the user initiates a retry and re-types the word.

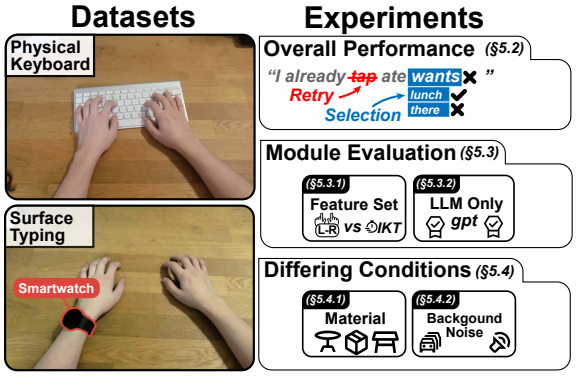


Figure 8: Figure depicts the setup for *Unikey*'s experiments. We collect two main datasets, one from each environment. The Physical Keyboard Dataset is primarily used to train *Unikey*, while the Surface Typing Dataset is used to evaluate *Unikey*. We evaluate *Unikey* overall effectiveness in word inference, as well as the effectiveness of the various system modules. Finally, we also evaluate the robustness of *Unikey* in different environments.

Ⓑ select the correct candidate from the list; or Ⓒ retry typing the word. This process repeats until the desired text is completed. Additionally, in the instances where a single word requires multiple retry attempts, *Unikey* blacklists words from each failed attempt, preventing them from appearing in the top N for subsequent retries until the correct word is selected. Finally, *Unikey* assumes words are typed accurately, but can rely on retries to handle any mistypes.

We consider *Unikey*'s primary performance indicator to be the amount of overhead introduced to the user while typing. In Ⓐ *Commit*, the overhead is minimal as the user briefly pauses to verify the correct result and continues typing. In Ⓑ *Select*, the user must choose the correct candidate, requiring additional time and effort. In Ⓒ *Retry*, the user retypes the entire word, which is the most time-consuming and effort-intensive action. Minimal frequency of *Select* and *Retry* actions indicate that *Unikey* performing effectively, with the user experiencing minimal overhead. In our experiments, we found that $N = 5$ minimizes *Retry* Rate while keeping the number of candidates that a user has to look through during each *Selection* reasonable. Finally, while the implementation of these actions lies outside the scope of this work, we address potential techniques for users to trigger such actions in the discussion section (see § 6).

5 Evaluation

We now present a comprehensive evaluation of *Unikey* to demonstrate its feasibility to support surface-based text inference.

5.1 Experimental Setup

Apparatus. We implement *Unikey* on a commercial smartwatch, specifically, the Samsung Galaxy Watch Active 2. We develop a Tizen 4.0 application on the watch to collect audio and IMU (accelerometer) traces. We build our signal processing and machine learning pipelines using Python libraries [62, 77, 87, 88], and utilize pre-trained models from Huggingface [112].

Participants. We recruit ten participants spread across both genders (six male, four female) from our university pool, within the age range of 20 to 35 years old, and regularly type as part of work or school requirements. Participants are proficient typists, with typical keyboard typing speeds ranging from 65 (Intermediate) to 114 (Advanced) words per minute (WPM), and use at least two different fingers of each hand when typing. We conduct our experiments adhering to our Institutional Review Board (IRB) approval.

Physical Keyboard Dataset. Physical keyboard keystrokes are required both to train *Unikey*, as well as evaluate the efficacy of *Unikey*'s baseline word-inference performance (i.e., without surface typing). We log keystrokes on physical keyboards from ten participants to create a dataset of their *Inter-Keystroke-Timings* (*IKTs*) on physical keyboards. We implement our keylogging tool, *Keystroke Collector*, using a Javascript/React web application similar to Monkeytype [79]. We ask the participants to use their personal computers to access *Keystroke Collector* and type the prompted text. For our experiments, we use the Oxford CEFR English A1 level Vocabulary [5] (891 words with homonyms removed), which has shown to be sufficient for basic daily communication [104]. Over a period of two months, we collect ten instances of *IKTs* for each word in the vocabulary, for a total of 89100 different data points (words) across all participants (excluding mistakes and rejected instances).

Surface Typing Dataset. To evaluate *Unikey*'s performance in a realistic scenario, we collect a smartwatch sensor dataset corresponding to the words typed on the surface of a wooden tabletop (see Figure 8). We collect data from all ten participants for a subset of words from the aforementioned CEFR A1 vocabulary, consisting of a total of 79 most frequently used unique words, stratified by word length (1-12 characters).

Before each collection session, we provide the participants with a visual guide in the form of a "printed paper keyboard" to type on, adhered to the tabletop [15, 110, 120], along with approximately

30 minutes of warm-up time to practice surface typing. Once familiarized, we collect the set of calibration taps required for *One-Time-Calibration* (see § 4.2.1), as well as smartwatch sensor data while participants type the predefined set of 79 words. We discuss the need for such a typing aid in § 6. To mitigate the effect of user typing error, we reject outlier word samples with a mean *IKT* that lies outside the three standard deviations range of the *Physical Keyboard* dataset, as well as incomplete or partially typed words, collecting at least five instances per vocabulary word from each participant. To increase test coverage, we re-combine and sample instances of sub-word tap features from this base vocabulary of 79 words to generate instances for an additional 98 vocabulary words, forming the final test vocabulary of 177 words, and a total of more than 9000 individual word instances across participants. Finally, for our evaluation, we form our *Test Sentences* of 1639 sentences (total word count of 10492 words) by sampling from the Corpora Typical Sentences [82] for sentences that contain only in-vocabulary words from the *Surface Typing* dataset.

Evaluation Metrics. Recall from Figure 7 that for each target word in a test sentence, after typing, *Unikey* outputs N candidate words (i.e., top N words). If the correct word does not appear in the top N, the word is considered an error and the user must Ⓒ *Retry* ($N = 5$) accordingly, repeating this process until the correct word appears in the top N. We define two key Word Error Rates (WER) as metrics to quantify *Unikey*'s *usability* as a text-entry system, namely ***Retry Rate (RR)*** which corresponds to Top-5 WER, and ***Selection Rate (SR)***, which correspond to the Top-1 WER after any *Retries*.

Specifically, let the total count of all words (i.e., sum of all sentence lengths) in *Test Sentences* be C_{words} and the count of words that have triggered the *Retry* process at least once be C_{retry}. We define ***Retry Rate (RR)*** as

$$RR = C_{retry}/C_{words}$$

Conversely, after any retry attempts (if any), if the correct word appears in the top N candidates, but not the first suggestion (i.e., Top-1 Result), the user must Ⓑ *select* the correct word. We denote the total count of such selection as C_{select}. The ***Selection Rate (SR)*** is then computed as

$$SR = C_{select}/C_{words}$$

We present WER as RR and SR as they quantify the additional overhead incurred when using *Unikey*, which is an indicator of *Unikey*'s text-inference performance. RR measures the chance that a word needs to be re-typed, while SR measures the overhead associated with finally selecting a word from the top N suggestions after all retry attempts. Lower values for both metrics indicate improved usability and text-inference performance.

5.2 *Unikey* Evaluation

Our primary investigation seeks to answer two questions: Can coarse-grained *spatial* and *temporal* information effectively infer the correct word in the context of wearable QWERTY surface-based text-entry (see § 5.2.1)? How effectively does *Unikey* transfer and apply typing patterns from a physical keyboard to the surface typing scenario (see § 5.2.2)?

These evaluations are conducted in an offline manner following the rules described in (§ 4.4) using real-world smartwatch data by sampling from either physical keyboard or surface typing datasets.

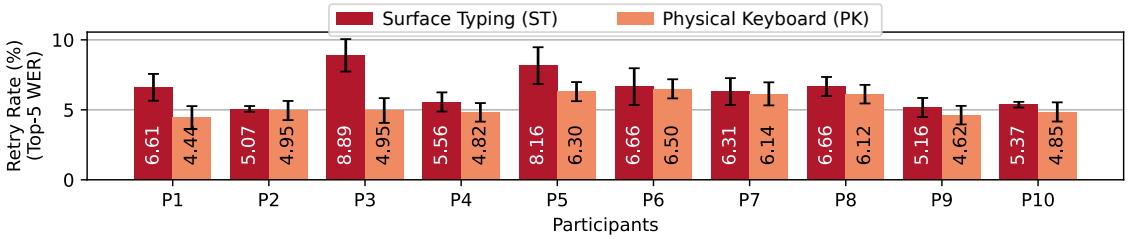
5.2.1 Physical Keyboard. We first evaluate *Unikey*'s performance in an "ideal" environment where typing patterns can be perfectly transferred from physical keyboards to surface-based typing (i.e., without noise introduced via sensors or environment). To do this, we test *Unikey* using samples from the physical keyboard dataset. Additionally, we ensure that *Unikey* training and testing sample sets are disjoint. It is important to note that despite the smaller test dictionary of 177 words, the word decoder model is trained on and searches the full CEFR A1 vocabulary (871 words) during inference.

As shown in Figure 9(a) and Figure 9(b), respectively, *Unikey* achieves an average *Retry Rate* of 5.369% and a *Selection Rate* of 29.76% with the "ideal" keyboard *IKTs*. In a similar context involving predictive auto-correction during mobile typing on a smartphone soft-keyboard, the average Word Error Rate (WER) after auto-correction (i.e., requiring the user to manually correct any mistakes) is approximately 13.4%. Additionally, the average rate at which users select words from the manual prediction panel is around 11% [20]. Our results indicate that by incorporating contextual information, "ideal" coarse-grained spatial and temporal word features can provide effective text inference on a wearable QWERTY keyboard. This approach introduces only a modest amount of corrective effort (in the form of word selections) for the user.

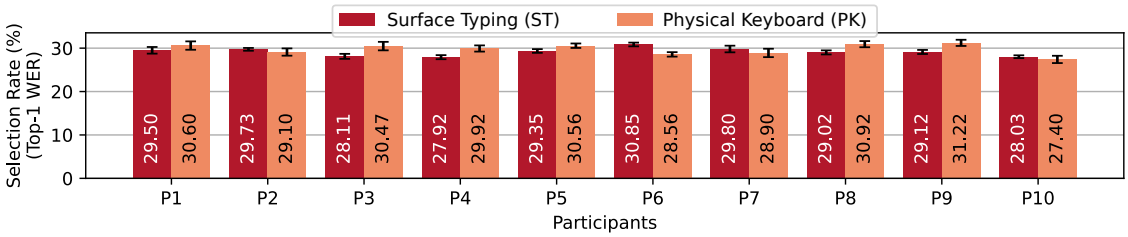
5.2.2 Surface-based Typing. To evaluate *Unikey* in a realistic setting that reflects its intended deployment, we train it on the physical keyboard dataset and test it on the surface typing dataset. The results in Figure 9 show that the performance of *Unikey* using the surface typing dataset as input is comparable to the previous experiment where the input is the physical keyboard dataset, achieving an average *Retry Rate* of 6.45% and *Selection Rate* of 29.14%. The best-performing participants achieve a competitive *Retry Rate* of 5.07% and a *Selection Rate* of 27.92%, respectively. There is an exception in the case of P3 and P5, which display relatively higher *Retry Rate*, but still similar *Selection Rate*. Further investigation reveals these participants occasionally used slightly altered finger assignments when typing on the surfaces as compared to a physical keyboard, resulting in different tap features, causing the *Word Classifier* to initially match with the wrong set of word candidates (primarily affecting *Retry Rate*). Since *Selection* only happens after all retries (wherein the correct candidate is guaranteed to appear in the top N), it depends primarily on semantic context instead of tap features. Therefore, the *Selection Rate* is not significantly affected.

Overall, the results indicate that *Unikey* can transfer the timing and localization information learned from a physical keyboard to a surface typing setting with high accuracy. Furthermore, since most participants only exhibit a slight increase in RR, given the aforementioned comparison with other mobile typing, these results indicate *Unikey*'s feasibility in practical scenarios.

5.2.3 Number of Retries. To provide a clearer idea of the cost overhead of *Retries*, we compute the mean cumulative distribution of retry attempts typing across all participants. In each scenario ("ideal" PK vs. practical ST), we observe that 98.2% (PK) and 97.5% (ST) of typing attempts can be resolved in at most one retry attempt, while



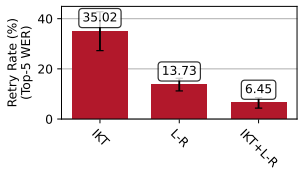
(a) Retry Rate of *Unikey*. This presents the likelihood that a user will have to retry a word. Lower values reflect more accurate inference.



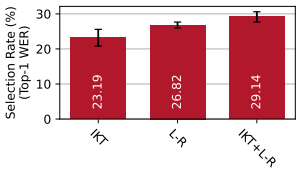
(b) Selection Rate of *Unikey*. This presents the likelihood that a user will be required to select from the top N list. Lower values reflect more accurate inference.

Figure 9: Figure depicts the overall performance of *Unikey*.

the 99th percentile requires at most three (PK) and four (ST) attempts, respectively. The results show that *Retries* are not only rare but also often can be resolved within just a few attempts, with only very few exceptional instances requiring a large number of retries. Upon further investigation, we found that these cases happen when the sensed *IKTs* L-R patterns match multiple words in the dictionary, and when the context is insufficient to disambiguate between similar words (i.e., start-of-sentence, multiple appropriate contextual candidates), presenting an extremely hard case of *Unikey*. However, given that the number of words even requiring retries is small (<6.45%), this occurs only very rarely in practice. Nevertheless, we discuss the potential mitigation of such cases in later sections (see § 6).



(a) Varying feature set on Retry Rate



(b) Varying feature set on Selection Rate

Figure 10: Figure depicts the effects of changing the feature set used by the Word Classifier.

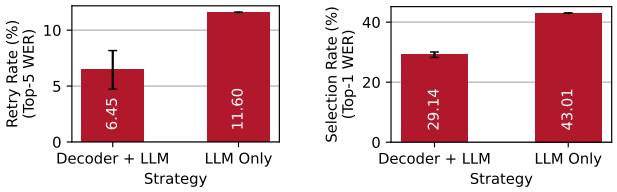
5.3 Verifying the Coarse-Grained Approach

In this investigation, we aim to understand the contribution of *Unikey*'s design choices and their implications on the effectiveness of *Unikey*'s core technique. We ask two key questions: Which features are most important for word classification (see § 5.3.1)? Can the LLM solve the search problem effectively on its own (see § 5.3.2)? We conduct our investigations using the surface typing dataset (i.e., practical scenario) and average results across users.

5.3.1 Effect of Feature Set. In this evaluation, we investigate the effect of changing the feature set used by *Word Classifier*. Recall that *Word Classifier* relies on the spatial L-R feature and temporal *IKT* feature for classification; we aim to find out how much each feature contributes to the overall performance of *Unikey*.

The results are shown in Figure 10. We observe that using only timing features (i.e., *IKT*) yields the worst performance in terms of *Retry Rate* (see Figure 10(a)), at 35.02%. Using spatial-only features (i.e., L-R), yields a significantly lower *Retry Rate* of 13.73%. This is expected, since *IKTs* alone are less descriptive than L-R, seeing as many completely unrelated key pairs can have similar *IKTs*, while the L-R feature guarantees at least the physical half of the keyboard each key was typed on. For *Selection Rate* (see Figure 10(b)), we observe a seemingly opposing trend, where using only timing features (i.e., *IKT*) yields the best performance of 23.19%, while combining features yields the worst performance at 29.14%. This is due to *Unikey*'s *Retry* mechanism blacklisting rejected candidates after each attempt, leading to an increased probability that subsequent candidates are inferred correctly.

Finally, the results show that combining both timing and spatial features (i.e., *IKT* + L-R) yields a much-improved performance in terms of *Retry Rate*, at 6.45%. This indicates that while the L-R features provide more information than timing, there is significant additional information that the timing feature can provide. Both are required for *Unikey*'s word inference.



(a) Retry Rate with and without the word classifier

(b) Selection Rate with and without the word classifier

Figure 11: Figure depicts the effect of *Unikey* with and without the word classifier.

5.3.2 Effect of using LLM without word classifier. As LLMs are known to be highly effective in text prediction, we investigate if the LLM can solve the search problem effectively on its own, without the top K words from the word classification component. We remove the word classification component and feed the output of the word feature extraction directly to the LLM model. Instead of the top K words, the LLM has to consider the entire *CEFR A1* dictionary, filtered by the appropriate word length.

The results are shown in Figure 11. We observe that without word classification to identify relevant words, *Unikey*'s performance drops significantly for both *Retry Rate* (see Figure 11(a)), which increases from 6.45% to 11.6%, and *Selection Rate* (see Figure 11(b)), which increases from 29.14% to 43.01%. The results show that the analysis of keystroke features and word classification contributes significantly to *Unikey*'s performance, and LLMs alone cannot solve the problem efficiently.

5.4 Impact of Environmental Conditions

We aim to understand environmental effects on *Unikey*'s typing by evaluating tap detection and word inference across different materials and background noise sources for a single user. We aim to show that our course-grained features are robust to such changes.
Tap Dataset. Using our smartwatch, we collect audio, IMU, and slow-motion video data of 8250 taps across 11 different surface materials for participant P1. The taps are evenly collected from typing the following randomly chosen subset of 5-character words from our test dictionary: 'there', 'would', 'place', 'right,' and 'thing'. Each word is typed 30 times, with the ground truth tap timing and L-R assignment manually annotated using the slow-motion video.

For Tap Detection, we define a true positive as a detected tap within 10ms of the ground truth tap, a false positive as a detected tap when there is no tap within 10ms, and a false negative as a missed tap. Let tp be the number of true positives, fp be the number of false positives, and fn be the number of false negatives. The tap detection accuracy is calculated as $tap_{acc} = \frac{tp_{tap}}{fp_{tap}+fn_{tap}+tp_{tap}}$. For L-R detection evaluation, we define a true positive, tp_{lr}, as a correct L-R detection and a false negative, fn_{lr}, as an incorrect L-R detection. The L-R detection accuracy is calculated as $lr_{acc} = \frac{tp_{lr}}{tp_{lr}+fn_{lr}}$.

5.4.1 Tap Detection on Different Surfaces. First, we evaluate 11 different tapping surfaces, as shown in Figure 12. *Tap Detection* has two components, (1) detection of a tap based on audio and (2) identification of L-R assignment based on IMU data. We perform our evaluations on a wide variety of common, everyday surfaces that could be used for *Unikey*. Surfaces can be split into three categories: *Flat* indicates a horizontal table-like surface, *On Lap* indicates a portable surface that can be placed on the lap, and *Vertical* indicates a vertical surface like a wall or a door.
Audio Tap Detection Performance. As shown in Figure 13, tap detection based on audio performs well on most surfaces, with an average accuracy of 94.4%. Surfaces like *WT*, *DM*, *HB*, and *CB*, achieve perfect accuracy due to their rigidity, which generates clear, impulsive tap sounds. Other rigid surfaces such as *MP*, *PB*, *CD*, and *IW* range above 97%, producing occasional taps that are either too soft due to awkward typing angle (*CD* and *IW*), flimsy structure (*PB*), or face interference from to loud and long-lived resonance (*MP*). Interestingly, *SB* performs well (97%) despite being soft, as it still produces significant noise. Surfaces like *GT* and *PL* show less than 90% accuracy. Unsurprisingly, *PL* suffers from indistinct tap sounds due to its softness. Though rigid, *GT* is made from hard glass, and also performs poorly as it produces soft noises that are hard to distinguish from the environment. Thus, *Unikey* generally works best on rigid surfaces that produce loud, impulsive sounds.
L-R assignment Performance. LR assignment works well on many surfaces, with an average accuracy of 95%. Hard and stable surfaces perform best, with high accuracy (>97%) achieved on flat surfaces like *WT*, *DM*, *MP*, and *GT*. Surfaces placed on the user's lap (e.g., *HB*, *CB*) also have high accuracy (100%) due to their large and rigid structural components. However, surfaces that are more flimsy and prone to shifting, like *PB*, have lower accuracy (93.3%). Similarly, softer surfaces like *SB* and *PL* also have lower accuracy (93.3% and 86%, respectively), since they create less impulsive tapping motions. Vertical surfaces like *CD* and *IW* perform poorly, likely due to awkward tapping angles resulting in less distinct IMU data.
Overall Performance. In addition to tap detection accuracy and LR assignment, similar to the differences between typing on keyboard and surface, typing on different surfaces might induce different degrees of variability in underlying typing behavior (i.e., changes in *IKTs*), which can affect further *Unikey*'s *Retry Rate*. Therefore, to investigate the combined impact of these factors on *Unikey*'s inference performance, we re-evaluate our primary experiment (see § 5.2.2) using the samples from the tap dataset. Figure 14 compares the mean *Retry Rate* of the five tap dataset words across the different surfaces. We consider the *Wooden Table* (*WT*), which is used in our primary experiment, as the baseline inference performance for these five words. For Flat and Lap surfaces with perfect tap detection accuracy (i.e., *DM*, *HB*, *CB*) and high L-R detection accuracy, we observe that *Retry Rate* remains low, and comparable with that of the *WT* baseline (8.62%), with a mean absolute difference of 0.78%. This indicates that on materials that allow for accurate tap detection and L-R assignment, typing patterns exhibit enough consistency for *Unikey*'s text-inference technique. Conversely, other surface materials (i.e., *MP*, *GT*, *PB*, *SB*, *PL*, *CD*, *IW*) suffer from an expected increase *Retry Rate* due to tap detection and L-R assignment errors. *GT* and *PL* are more severe examples, with a corresponding 72.08% and 52.04% increase compared to *WT* baseline, respectively, while the remaining materials exhibit much more minor increases ranging between an additional 4.78% to 22.6%.

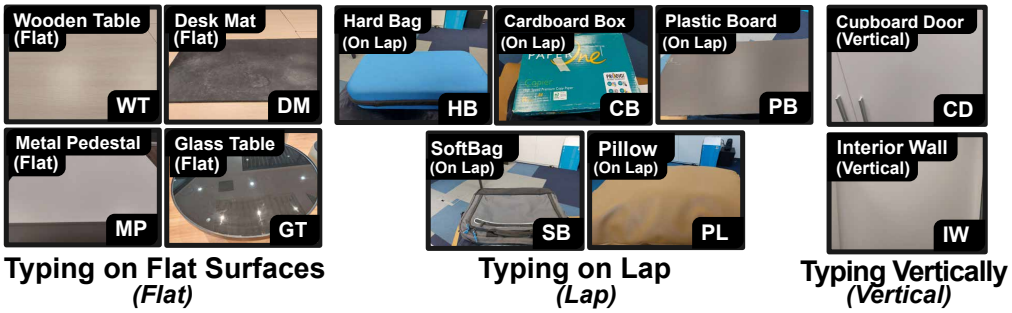


Figure 12: Figure depicts the different materials used to evaluate *Unikey*'s tap detection performance.

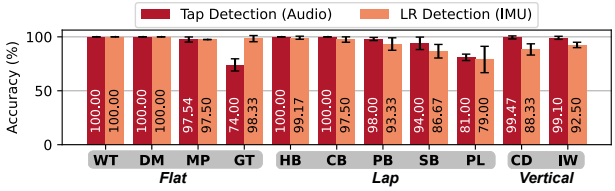


Figure 13: Figure depicts *Unikey*'s tap detection performance on different materials.

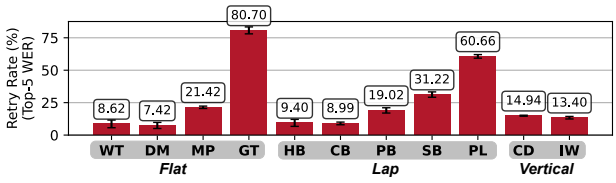


Figure 14: Figure depicts the average *Retry Rate* (i.e., Top-5 WER) of the words in the Tap Dataset when typed on different materials.

Notably, despite suffering from reduced L-R assignment accuracy (10% reduction), Vertical Surfaces such as *CD* and *IW* only exhibit about a 5% increase in *Retry Rate*, which suggests that performance can be further improved in such situations with more complex L-R detection technique. Finally, while these surfaces result in reduced performance for *Unikey*, we note that stable and rigid surfaces in which *Unikey* works well (i.e., desks, counter-tops) can be found in many typical everyday spaces (i.e., homes, offices) or on the go (i.e., hard carrying cases, briefcases), and would allow users to still use *Unikey* effectively in many situations.

5.4.2 Background Noise on Audio Tap Detection. We evaluate 12 different types of background noise at different SNRs on the Wooden Table (WT) surface. We present background noise types in three groups; indoor environmental noises (i.e., *Cafe, Office, Kitchen, Speech, Distant Typing*), outdoor environmental noises (i.e., *Park, Airport, Station, Bus, Metro*), and cases we find to be particularly adverse for *Unikey*'s pipeline (i.e., *Traffic, Nearby Typing*). We digitally mix the audio with different background noise types at SNR levels +/-20dB relative to the power of the original audio signal.

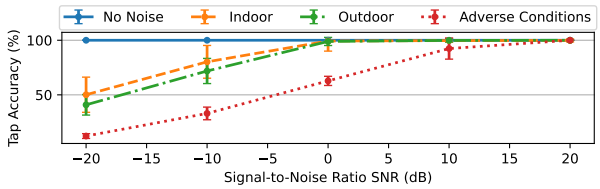


Figure 15: Figure depicts accuracy of *Unikey*'s audio tap detection on different background types and Signal to Noise Ratios. No Noise denotes baseline audio, Indoor noises denote more private indoor situations (i.e., Cafe, Office, Kitchen, Speech, Distant Typing) and Outdoor denotes more public, outdoor spaces (i.e., Park, Airport, Station, Bus, Metro). Adverse Conditions denote background noises that are particularly disruptive to *Unikey* tap detection.

As shown in Figure 15, tap detection accuracy improves with increasing Signal to Noise Ratio (SNR). In typical use, taps are recorded near the smartwatch, away from noise sources, resulting in positive SNRs and strong performance across different background noises. At 0dB SNR, most materials maintain high accuracy (i.e., greater than 99%), except for Traffic and Typing Near (i.e., Adverse Conditions), which drop to as low as 60% due to low-frequency, impulsive noises causing significantly more false positives. Notably, although not directly shown in the figure, we note that even at -10dB SNR, *Unikey* remains resilient to indoor noises like Cafe, Office, and Kitchen, maintaining high accuracy of 95%, 98%, and 99%, respectively. Given that taps are made close to the smartwatch, we can reasonably expect that taps are at least as loud as the background (0dB SNR), which allows *Unikey* to be usable in most indoor and outdoor scenarios.

6 Discussion

We discuss the deployment considerations, limitations, and future work for *Unikey*.

6.1 Considerations For Practical Deployment

We discuss the steps and components needed to deploy *Unikey* as a usable text-entry system, from its current prototype status.

Handling Non-Word Inputs. With reference to § 4.4, aside from text-entry (i.e., words), *Unikey* also requires some input method

to trigger *Retries* and to perform *Selections*, which we currently do not provide. Since the number of such actions is small, we envision that this can be implemented "out-of-band" using gesture engines already present in many smartwatches [12–14]. These gestures are often simple and fast, intended for quickly navigating through and selecting different menu items without interacting with the touchscreen [12], which adapts well to *Unikey*'s small set of non-typing input actions (i.e., Retry, Select Candidate *N*). In cases where *Unikey* is used together with head-mounted wearables (i.e., AR/VR smart glasses, earables), this can alternatively be implemented using gaze-directed or head-motion based input [19, 116]. Additionally, other non-word keyboard inputs (i.e., caps-lock, tab) can also be represented using such gestures, further extending *Unikey*'s functionality. Particularly, replacing the current delay-based word delineation with a quick "space-bar" gesture could significantly improve *Unikey*'s typing speed in an actual deployment.

Visual Layout Guide. While *Unikey* is designed for use without additional accessories, some participants found it unfamiliar to type without a visual layout guide. To ease this transition, we provide a standard QWERTY "printed paper keyboard" as a reference to help users orient themselves to *Unikey*'s surface-based typing (See § 5.2.2). Although carrying a foldable paper keyboard introduces a minor inconvenience, prior studies suggest that many users are already proficient at typing on physical keyboards without looking down at the keys [89]. Over time, we anticipate that users will develop a similar level of comfort with surface typing without requiring the assistance of printed paper keyboard. For those who may require additional guidance, the keyboard layout could also be projected onto the typing surface using AR or VR displays.

UI and Feedback. We envision that *Unikey* runs in conjunction with larger system that can provide visual feedback to the user (i.e., SmartTVs, AR/VR headsets), connected via wireless networking (e.g., WiFi, Bluetooth), with the more computationally intensive parts of the pipeline (i.e., LLM) running on the more powerful host machine (i.e., not the smartwatch). In our offline testing of *Unikey*, we encountered an average of 18ms processing delay for each word (LLM included), which leaves ample room for network delays to achieve real-time display feedback. Furthermore, since *Unikey*'s user flow is comparable to existing widely used IMEs (see § 4.4), we envision that a similar display would work well for our approach. A key advantage of *Unikey* is that users type individual words at near-regular keyboard speed, as the core input mimics familiar typing behavior. Most time overhead arises from retries, selections, and delineating between words. Using actual measured IKTs (100-200ms/char), selections and retry rates, and approximating 0.5s for word separation and 1s selection latency[20], we estimate that such a system can achieve end-to-end typing speeds of 45-50 WPM, comparable to many similar XR text entry techniques [26]. We leave the implementation, design and benchmarking of an optimal UI for users, as future work.

6.2 Limitations

Retries. While *Unikey* achieves a low *Retry Rate*, there are rare instances where users may need to attempt a word several times before succeeding, which could momentarily affect the typing experience. These cases typically arise due to ambiguities in both tap features and semantic context. While some ambiguity in tap features is unavoidable due to their coarse-grained nature, our current system does not yet incorporate chat history for context, which could significantly improve disambiguation. With the integration of chat history, *Unikey* could further reduce the need for retries, enhancing accuracy and user experience. In situations where sufficient context is still unavailable, users can briefly switch to an alternative input method (e.g., a smartwatch on-screen keyboard) to complete the word, maintaining a fluid and reliable typing experience.

Reliance on Consistent Typing Habits. Unlike a physical keyboard, where even novices can press the correct keys, highly irregular timing or finger usage introduces excessive noise in both *IKTs* and L-R features. Therefore, *Unikey* can struggle with individuals who have not formed consistent habits (e.g., single-finger, single-hand typists, learner typists). While we acknowledge that some users are ultimately unable to use *Unikey*, prior research shows that many typists, both self-taught or touch typists, eventually converge on efficient finger movement strategies [45], which are precisely the patterns we leverage in our design. Since these patterns do not need to be perfectly distinct, but rather just sufficient to reduce the search-space of possible words to a manageable size for the LLM to disambiguate, we believe that *Unikey* can generalize well as long as the users type on a regular basis. Furthermore, we envision that *Unikey* can update itself over time, using both typing data from regular keyboard use and from *Unikey* itself, adapting as the user becomes more proficient or typing behavior changes.

Generalization of Typing Surface Results. The typing surface study (see § 5.4) used a single participant (P1); our goal was to isolate the effects of material properties (e.g., acoustic response, stability, rigidity) while controlling other factors. Since our primary study shows that most users exhibit enough consistency on the wooden table for *Unikey* to provide accurate inference, we expect that within reasonable bounds for typing force, the other stable and rigid typing surfaces with good acoustic characteristics that provide a similar typing experience, would also generalize well to other users. However, we acknowledge that user-specific variations, especially with more exotic typing surfaces, can play a significant role in accuracy. We plan to expand this study in future work.

Limited Dictionary Size. To limit the scale of our experiments, *Unikey* is evaluated with a limited CEFR A1 vocabulary of around 900 words, which vocabulary is already sufficient for basic communication tasks (e.g., asking for directions, expressing needs). A slightly larger B1 vocabulary (i.e., around 2750 words) would allow *Unikey* to scale to most informal day-to-day contexts and conversations [5, 78]. To account for the expected increase in ambiguous words with larger vocabularies, the LLM could be fine-tuned to fit to the user's writing style, which can be accomplished alongside learning typing behavior. Furthermore, in chat-like applications, the context can be extended to include previous messages, which would further aid in disambiguation. While *Unikey* might not be scale to complex typing tasks (e.g., writing essays, technical reports), we envision that *Unikey* can be extended to minimally support a user's lexicon of commonly-used words, which would be sufficient for many everyday situations (i.e., replying to messages, searching the web, communicating with AI assistants) where surface typing would provide the most convenience.

Passwords. Currently, *Unikey* is unable to handle passwords because it classifies words from a set vocabulary. However, there have been other studies exploring the use of *Inter-Keystroke-Timings* for user authentication. Since *Inter-Keystroke-Timings* are somewhat unique to each user [40] and passwords are often typed with frequency and familiarity [86], we envision *Unikey* learning these patterns for authentication purposes in future iterations. Furthermore, mobile and wearable-friendly authentication methods (i.e., handwriting signatures [67, 69], biometrics [48, 65]) can work alongside *Unikey* to provide alternatives to passwords in such scenarios.

6.3 Future of *Unikey*

In this work, we focus on demonstrating *Unikey*'s core technique on a smartwatch as a proof of concept. While we already indicate key directions for implementing *Unikey* as a full-fledged *smartwatch-based* text-entry system, we note that our approach, or *Unikey*'s primary features (*IKTs* and L-R), can be adapted to other devices (i.e., smartphones and smart speakers) with minimal modification. Since *Unikey* relies on audio-based tap detection, it can be implemented on any device equipped with a microphone. Additionally, prior research [110] demonstrates the feasibility of fine-grained keystroke localization using just audio, suggesting that *Unikey* could extract coarse-grained L-R information with minimal overhead. This opens the possibility of implementing *Unikey* using audio alone, making it deployable across a wide range of commercial off-the-shelf (COTS) devices. In fact, we envision that an ideal version of *Unikey* would work with any sensor that can provide our coarse-grained features.

With advancements in AI and LLMs, we expect that future text-entry systems would be able to take full advantage of not just textual information, but also information from the user's environment (i.e., visual, audio, current activity, application context) [35, 49, 57, 68]. Harnessing this could lead to even less reliance on fine-grained or high resolution sensor data, requiring only minimal and vague sensor input as a reference for contextual inference of the user's intended text. Our work is a step towards this future, where we demonstrate that even a small amount of coarse-grained sensor data can be used to infer text-input with high accuracy.

7 Conclusion

We present a novel approach to text-entry on surfaces that combines coarse-grained temporal and spatial information with the power of large language models (LLMs) to enable accurate text inference. Our prototype system, *Unikey*, demonstrates the feasibility of this approach using only a single commodity smartwatch, offering the potential for an out-of-the-box solution for mobile technologies such as augmented and virtual reality (AR/VR), wearables, and smart TVs, where traditional keyboards are impractical.

References

[1] 2014. Improving Smart TV Security with Simpler and Stronger Authentication. https://www.secsign.com/solving-smart-tv-security-simpler-stronger-authentication/

[2] 2019. What comes after the keyboard? https://www.verizon.com/about/our-company/fourth-industrial-revolution/what-comes-after-keyboard

[3] 2022. Virtual Reality And The Input Problem. https://www.bruceb.com/2022/10/virtual-reality-and-the-input-problem/

[4] 2024. Input method (IME) – Google Input Tools. https://www.google.com/inputtools/services/features/input-method.html

[5] 2024. International language standards | Cambridge English. https://www.cambridgeenglish.org/exams-and-tests/cefr/

[6] 2024. Microsoft Simplified Chinese IME - Microsoft Support. https://support.microsoft.com/en-us/windows/microsoft-simplified-chinese-ime-9b962a3b-2fa4-4f37-811c-b1886320dd72

[7] 2024. Oculus Quest: All-in-One VR Headset | Oculus. https://www.oculus.com/quest/refurbished/

[8] 2024. RayNeo Air2 AR Glasses. https://www.rayneo.com/products/rayneo-air-2-xr-glasses

[9] 2024. Virtual Keyboard for Smart TV and Internet Video. https://www.softcorporation.com/smarttv/

[10] 2024. XREAL Air | Elevate Your AR Experience with Cutting-Edge Smart Glasses. https://www.xreal.com/

[11] 2025. Even Realities G1 Review | Prescription Smart Glasses with Display & AI. https://www.evenrealities.com/g1

[12] 2025. More Than a Gesture: How Galaxy Watch's Universal Gestures Feature Enhances Accessibility. https://news.samsung.com/global/more-than-a-gesture-how-galaxy-watchs-universal-gestures-feature-enhances-accessibility

[13] 2025. Navigate your watch with wrist gestures - Wear OS by Google Help. https://support.google.com/wearos/answer/6312406?hl=en

[14] 2025. Use AssistiveTouch on Apple Watch. https://support.apple.com/en-sg/guide/watch/apdec70bfd2d/watchos

[15] Heba Abdelnasser, Khaled A. Harras, and Moustafa Youssef. 2020. MagStroke: A Magnetic Based Virtual Keyboard for Off-the-Shelf Smart Devices. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 1–9. https://doi.org/10.1109/SECON48991.2020.9158416 ISSN: 2155-5494.

[16] Saleh M. Abu-Soud. 2016. PaSSIL: A New Keystroke Dynamics System for Password Strengthening Based on Inductive Learning. https://www.semanticscholar.org/paper/PaSSIL%3A-A-New-Keystroke-Dynamics-System-for-Based-Abu-Soud/707ff05fce455a3e21c7c62273cc44a8cfb22683

[17] Alejandro Acien, Aythami Morales, John V. Monaco, Ruben Vera-Rodriguez, and Julian Fierrez. 2022. TypeNet: Deep Learning Keystroke Biometrics. *IEEE Transactions on Biometrics, Behavior, and Identity Science* 4, 1 (Jan. 2022), 57–70. https://doi.org/10.1109/TBIOM.2021.3112540 Conference Name: IEEE Transactions on Biometrics, Behavior, and Identity Science.

[18] Alejandro Acien, Aythami Morales, Ruben Vera-Rodriguez, Julian Fierrez, and John V. Monaco. 2020. TypeNet: Scaling up Keystroke Biometrics. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*. 1–7. https://doi.org/10.1109/IJCB48548.2020.9304908 ISSN: 2474-9699.

[19] Sunggeun Ahn and Geehyuk Lee. 2019. Gaze-Assisted Typing for Smart Glasses. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. Association for Computing Machinery, New York, NY, USA, 857–869. https://doi.org/10.1145/3332165.3347883

[20] Ohoud Alharbi, Wolfgang Stuerzlinger, and Felix Putze. 2020. The Effects of Predictive Features of Mobile Keyboards on Text Entry Speed and Errors. *Proc. ACM Hum.-Comput. Interact.* 4, ISS (Nov. 2020), 183:1–183:16. https://doi.org/10.1145/3427311

[21] A. AZZALINI and A. DALLA VALLE. 1996. The multivariate skew-normal distribution. *Biometrika* 83, 4 (Dec. 1996), 715–726. https://doi.org/10.1093/biomet/83.4.715

[22] Jia-Xuan Bai, Bin Liu, and Luchuan Song. 2021. I Know Your Keyboard Input: A Robust Keystroke Eavesdropper Based-on Acoustic Signals. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21)*. Association for Computing Machinery, New York, NY, USA, 1239–1247. https://doi.org/10.1145/3474085.3475539

[23] Nick Bartlow and Bojan Cukic. 2009. Keystroke Dynamics-Based Credential Hardening Systems. In *Handbook of Remote Biometrics: for Surveillance and Security*, Massimo Tistarelli, Stan Z. Li, and Rama Chellappa (Eds.). Springer, London, 329–347. https://doi.org/10.1007/978-1-84882-385-3_14

[24] Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. 2002. User authentication through keystroke dynamics. *ACM Trans. Inf. Syst. Secur.* 5, 4 (Nov. 2002), 367–397. https://doi.org/10.1145/581271.581272

[25] Bhaveer Bhana and Stephen Flowerday. 2020. Passphrase and keystroke dynamics authentication: Usable security. *Computers & Security* 96 (Sept. 2020), 101925. https://doi.org/10.1016/j.cose.2020.101925

[26] Arpit Bhatia, Moaaz Hudhud Mughrabi, Diar Abdlkarim, Massimiliano Di Luca, Mar Gonzalez-Franco, Karan Ahuja, and Hasti Seifi. 2025. Text Entry for XR Trove (TEXT): Collecting and Analyzing Techniques for Text Input in XR. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, 1–18. https://doi.org/10.1145/3706598.3713382

[27] M. K. Bhuyan, D. Ajay Kumar, Karl F. MacDorman, and Yuji Iwahori. 2014. A novel set of features for continuous hand gesture recognition. *Journal on Multimodal User Interfaces* 8, 4 (Oct. 2014), 333–343. https://doi.org/10.1007/s12193-014-0165-0

[28] Patrick Bours and Hafez Barghouthi. 2009. Continuous Authentication using Biometric Keystroke Dynamics. (2009).

[29] Matt Burgess. 2024. I Stopped Using Passwords. It's Great—and a Total Mess | WIRED. https://www.wired.com/story/stopped-using-passwords-passkeys/

[30] Pieter Buzing. 2003. Comparing Different Keyboard Layouts: Aspects of QWERTY, DVORAK and alphabetical keyboards. (Jan. 2003).

[31] Stefano Cecconello, Alberto Compagno, Mauro Conti, Daniele Lain, and Gene Tsudik. 2019. Skype & Type: Keyboard Eavesdropping in Voice-over-IP. *ACM Trans. Priv. Secur.* 22, 4 (Dec. 2019), 24:1–24:34. https://doi.org/10.1145/3365366

[32] Long Chen, Xianchao Wu, and Jingzhou He. 2012. Using Collocations and K-means Clustering to Improve the N-pos Model for Japanese IME. In *Proceedings of the Second Workshop on Advances in Text Input Methods*, Kalika Bali, Monojit Choudhury, and Yoh Okuno (Eds.). The COLING 2012 Organizing Committee, Mumbai, India, 45–56. https://aclanthology.org/W12-4804

[33] Siyu Chen, Hongbo Jiang, Jingyang Hu, Zhu Xiao, and Daibo Liu. 2024. Silent Thief: Password Eavesdropping Leveraging Wi-Fi Beamforming Feedback from POS Terminal. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*. 321–330. https://doi.org/10.1109/INFOCOM52122.2024.10621321 ISSN: 2641-9874.

[34] Sibo Chen, Junce Wang, Santiago Guerra, Neha Mittal, and Soravis Prakkamakul. 2019. Exploring Word-gesture Text Entry Techniques in Virtual Reality. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (CHI EA '19)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3290607.3312762

[35] Xiangxiang Chu, Jianlin Su, Bo Zhang, and Chunhua Shen. 2024. VisionLLaMA: A Unified LLaMA Backbone for Vision Tasks. https://doi.org/10.48550/arXiv.2403.00522 arXiv:2403.00522 [cs].

[36] Gennaro Costagliola, Vittorio Fuccella, and Michele Di Capua. 2010. Interpreting Gestures for Text Entry on Touch Screen Devices. 315–320.

[37] S.T. de Magalhaes, K. Revett, and H.M.D. Santos. 2005. Password secured sites - stepping forward with keystroke dynamics. In *International Conference on Next Generation Web Services Practices (NWeSP'05)*. 6 pp.–. https://doi.org/10.1109/NWESP.2005.62

[38] Yunbin Deng and Yu Zhong. 2015. Keystroke Dynamics User Authentication Using Advanced Machine Learning Methods. In *Gate to Computer Science and Research* (1st ed.), Yu Zhong and Yunbin Deng (Eds.). Vol. 2. Science Gate Publishing P.C., 23–40. https://doi.org/10.15579/gcsr.vol2.ch2

[39] 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*. ACM, Montreal QC Canada, 1–12. https://doi.org/10.1145/3173574.3174220

[40] 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 (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3174220

[41] William B. Dress. 1997. Applications of a fast continuous wavelet transform. In *Wavelet Applications IV*, Vol. 3078. SPIE, 570–580. https://doi.org/10.1117/12.271748

[42] H. Du, T. Oggier, F. Lustenberger, and E. Charbon. 2005. A Virtual Keyboard Based on True-3D Optical Ranging. In *Procedings of the British Machine Vision Conference 2005*. British Machine Vision Association, Oxford, 27.1–27.10. https://doi.org/10.5244/C.19.27

[43] Saba Fallah and Scott Mackenzie. 2023. H4VR: One-handed Gesture-based Text Entry in Virtual Reality Using a Four-key Keyboard. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems (CHI EA '23)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3544549.3585876

[44] Yuyi Fang, Zhiwei Zhao, Zi Wang, Geyong Min, Yue Cao, Haojun Huang, and Hao Yin. 2018. Eavesdrop with PoKeMon: Position free keystroke monitoring using acoustic data. *Future Generation Computer Systems* 87 (Oct. 2018), 704–711. https://doi.org/10.1016/j.future.2017.10.039

[45] 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 (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4262–4273. https://doi.org/10.1145/2858036.2858233

[46] Zongkai Fu, Huiyong Li, Zhenchao Ouyang, Xuefeng Liu, and Jianwei Niu. 2020. Typing Everywhere with an EMG Keyboard: A Novel Myo Armband-Based HCI Tool. In *Algorithms and Architectures for Parallel Processing*, Meikang Qiu (Ed.).

Springer International Publishing, Cham, 247–261. https://doi.org/10.1007/978-3-030-60245-1_17

[47] P.P. Gandhi and S.A. Kassam. 1994. Optimality of the cell averaging CFAR detector. *IEEE Transactions on Information Theory* 40, 4 (July 1994), 1226–1228. https://doi.org/10.1109/18.335950 Conference Name: IEEE Transactions on Information Theory.

[48] Yang Gao, Wei Wang, Vir V. Phoha, Wei Sun, and Zhanpeng Jin. 2019. EarEcho: Using Ear Canal Echo for Wearable Authentication. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 3 (Sept. 2019), 81:1–81:24. https://doi.org/10.1145/3351239

[49] Zhiqi Ge, Hongzhe Huang, Mingze Zhou, Juncheng Li, Guoming Wang, Siliang Tang, and Yueting Zhuang. 2024. WorldGPT: Empowering LLM as Multimodal World Model. In *Proceedings of the 32nd ACM International Conference on Multimedia (MM '24)*. Association for Computing Machinery, New York, NY, USA, 7346–7355. https://doi.org/10.1145/3664647.3681488

[50] Romain Giot, Mohamad El-Abed, Baptiste Hemery, and Christophe Rosenberger. 2011. Unconstrained keystroke dynamics authentication with shared secret. *Computers & Security* 30, 6 (Sept. 2011), 427–445. https://doi.org/10.1016/j.cose.2011.03.004

[51] Romain Giot, Mohamad El-Abed, and Christophe Rosenberger. 2009. Keystroke dynamics authentication for collaborative systems. In *2009 International Symposium on Collaborative Technologies and Systems*. 172–179. https://doi.org/10.1109/CTS.2009.5067478

[52] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. 2011. OmniTouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST '11)*. Association for Computing Machinery, New York, NY, USA, 441–450. https://doi.org/10.1145/2047196.2047255

[53] Zhenyi He, Christof Lutteroth, and Ken Perlin. 2022. TapGazer: Text Entry with Finger Tapping and Gaze-directed Word Selection. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1–16. https://doi.org/10.1145/3491102.3501838

[54] Alex Hern. 2017. I tried to work all day in a VR headset and it was horrible. *The Guardian* (Jan. 2017). https://www.theguardian.com/technology/2017/jan/05/i-tried-to-work-all-day-in-a-vr-headset-so-you-never-have-to

[55] Sylvain Hocquet, Jean-Yves Ramel, and Hubert Cardot. 2007. User Classification for Keystroke Dynamics Authentication. In *Advances in Biometrics*, Seong-Whan Lee and Stan Z. Li (Eds.). Springer, Berlin, Heidelberg, 531–539. https://doi.org/10.1007/978-3-540-74549-5_56

[56] Jingyang Hu, Hongbo Wang, Tianyue Zheng, Jingzhi Hu, Zhe Chen, Hongbo Jiang, and Jun Luo. 2023. Password-Stealing without Hacking: Wi-Fi Enabled Practical Keystroke Eavesdropping. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*. Association for Computing Machinery, New York, NY, USA, 239–252. https://doi.org/10.1145/3576915.3623088

[57] Wenbo Hu, Yifan Xu, Yi Li, Weiyue Li, Zeyuan Chen, and Zhuowen Tu. 2024. BLIVA: A Simple Multimodal LLM for Better Handling of Text-Rich Visual Questions. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 3 (March 2024), 2256–2264. https://doi.org/10.1609/aaai.v38i3.27999 Number: 3.

[58] Yuqian Hu, Beibei Wang, Chenshu Wu, and K.J. Ray Liu. 2022. mmKey: Universal Virtual Keyboard Using A Single Millimeter-Wave Radio. *IEEE Internet of Things Journal* 9, 1 (Jan. 2022), 510–524. https://doi.org/10.1109/JIOT.2021.3084560

[59] Jack. 2023. Next-Level Convenience: Smart and IoT Authentication Simplifies Login on Smart TVs, Game Consoles. https://medium.com/@jackforbesJF/next-level-convenience-smart-and-iot-authentication-simplifies-login-on-smart-tvs-game-consoles-c25a34db74f4

[60] Suman Jana and Vitaly Shmatikov. 2012. Memento: Learning Secrets from Process Footprints. In *2012 IEEE Symposium on Security and Privacy*. IEEE, San Francisco, CA, USA, 143–157. https://doi.org/10.1109/SP.2012.19

[61] Suman Jana and Vitaly Shmatikov. 2012. Memento: Learning Secrets from Process Footprints. In *2012 IEEE Symposium on Security and Privacy*. IEEE, San Francisco, CA, USA, 143–157. https://doi.org/10.1109/SP.2012.19

[62] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001. SciPy: Open source scientific tools for Python. http://www.scipy.org/

[63] Jung In Koh, Josh Cherian, Paul Taele, and Tracy Hammond. 2019. Developing a Hand Gesture Recognition System for Mapping Symbolic Hand Gestures to Analogous Emojis in Computer-Mediated Communication. *ACM Trans. Interact. Intell. Syst.* 9, 1 (March 2019), 6:1–6:35. https://doi.org/10.1145/3297277

[64] Adrienne LaFrance. 2016. The Quest For the Next Human-Computer Interface. https://www.theatlantic.com/technology/archive/2016/07/what-comes-after-keyboards/490511/ Section: Technology.

[65] Sunwoo Lee, Wonsuk Choi, and Dong Hoon Lee. 2021. Usable User Authentication on a Smartwatch using Vibration. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*. Association for Computing Machinery, New York, NY, USA, 304–319. https://doi.org/10.1145/3460120.3484553

[66] Seng-Hong Lee, ZiXun Yu, KeKe Chen, and Xiao Li. 2024. KeyAtNet: Keystroke Signal Real-time Eavesdropping Based on Fourier Neural Operator. In *Advances in Neural Networks – ISNN 2024*, Xinyi Le and Zhijun Zhang (Eds.). Springer Nature, Singapore, 288–297. https://doi.org/10.1007/978-981-97-4399-5_27

[67] Alona Levy, Ben Nassi, Yuval Elovici, and Erez Shmueli. 2018. Handwritten Signature Verification Using Wrist-Worn Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3 (Sept. 2018), 119:1–119:26. https://doi.org/10.1145/3264929

[68] Dongting Li, Chenchong Tang, and Han Liu. 2024. Audio-LLM: Activating the Capabilities of Large Language Models to Comprehend Audio Data. In *Advances in Neural Networks – ISNN 2024*, Xinyi Le and Zhijun Zhang (Eds.). Springer Nature, Singapore, 133–142. https://doi.org/10.1007/978-981-97-4399-5_13

[69] Gen Li and Hiroyuki Sato. 2020. Handwritten Signature Authentication Using Smartwatch Motion Sensors. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. 1589–1596. https://doi.org/10.1109/COMPSAC48688.2020.00-28 ISSN: 0730-3157.

[70] Changsung Lim, Jina Kim, and Myung Jin Kim. 2022. Thumble: One-Handed 3D Object Manipulation Using a Thimble-Shaped Wearable Device in Virtual Reality. In *Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 1–3. https://doi.org/10.1145/3526114.3558703

[71] Xinye Lin, Yixin Chen, Xiao-Wen Chang, Xue Liu, and Xiaodong Wang. 2018. SHOW: Smart Handwriting on Watches. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4 (Jan. 2018), 151:1–151:23. https://doi.org/10.1145/3161412

[72] Moritz Lipp, Daniel Gruss, Michael Schwarz, David Bidner, Clémentine Maurice, and Stefan Mangard. 2017. Practical Keystroke Timing Attacks in Sandboxed JavaScript. 191–209. https://doi.org/10.1007/978-3-319-66399-9_11

[73] Jian Liu, Yingying Chen, and Marco Gruteser. 2016. VibKeyboard: Virtual Keyboard Leveraging Physical Vibration: Demo. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking (MobiCom '16)*. Association for Computing Machinery, New York, NY, USA, 507–508. https://doi.org/10.1145/2973750.2985624

[74] Jian Liu, Yan Wang, Gorkem Kar, Yingying Chen, Jie Yang, and Marco Gruteser. 2015. Snooping Keystrokes with mm-level Audio Ranging on a Single Phone. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*. Association for Computing Machinery, New York, NY, USA, 142–154. https://doi.org/10.1145/2789168.2790122

[75] Khalid Mansour and Khaled Mahmoud. 2021. A New Approach for Textual Password Hardening Using Keystroke Latency Times. *The International Arab Journal of Information Technology* 18, 3 (May 2021). https://doi.org/10.34028/iajit/18/3/10

[76] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. 2011. (sp)iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, Chicago Illinois USA, 551–562. https://doi.org/10.1145/2046707.2046771

[77] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, Vol. 8.

[78] James Milton and Thomai Alexiou. 2009. *Vocabulary Size and the Common European Framework of Reference for Languages*. 194–211. https://doi.org/10.1057/9780230242258_12

[79] Miodec. 2025. Monkeytype | A minimalistic, customizable typing test. https://monkeytype.com/

[80] Fabian Monrose, Michael K. Reiter, and Susanne Wetzel. 1999. Password hardening based on keystroke dynamics. In *Proceedings of the 6th ACM conference on Computer and communications security (CCS '99)*. Association for Computing Machinery, New York, NY, USA, 73–82. https://doi.org/10.1145/319709.319720

[81] Fabian Monrose and Aviel Rubin. 1997. Authentication via keystroke dynamics. In *Proceedings of the 4th ACM conference on Computer and communications security*. ACM, Zurich Switzerland, 48–56. https://doi.org/10.1145/266420.266434

[82] Lydia Müller, Uwe Quasthoff, and Maciej Sumalvico. 2018. Corpora of Typical Sentences. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga (Eds.). European Language Resources Association (ELRA), Miyazaki, Japan. https://aclanthology.org/L18-1688

[83] Shahriar Nirjon, Jeremy Gummeson, Dan Gelb, and Kyu-Han Kim. 2015. TypingRing: A Wearable Ring Platform for Text Input. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '15)*. Association for Computing Machinery, New York, NY, USA, 227–239. https://doi.org/10.1145/2742647.2742665

[84] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. 2012. ACCessory: Password Inference Using Accelerometers on Smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications (HotMobile '12)*. Association for Computing Machinery, New York, NY, USA,

1–6. https://doi.org/10.1145/2162081.2162095

[85] Sourav Panda, Yuanzhen Liu, Gerhard Petrus Hancke, and Umair Mujtaba Qureshi. 2020. Behavioral Acoustic Emanations: Attack and Verification of PIN Entry Using Keypress Sounds. *Sensors* 20, 11 (Jan. 2020), 3015. https://doi.org/10.3390/s20113015 Number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

[86] Simon Parkinson, Saad Khan, Alexandru-Mihai Badea, Andrew Crampton, Na Liu, and Qing Xu. 2023. An empirical analysis of keystroke dynamics in passwords: A longitudinal study. *IET Biometrics* 12, 1 (2023), 25–37. https://doi.org/10.1049/bme2.12087 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/bme2.12087.

[87] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. https://doi.org/10.48550/arXiv.1912.01703 arXiv:1912.01703 [cs, stat].

[88] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[89] Svetlana Pinet. 2024. What are you looking at? Beyond typing speed and formal training for assessing typing expertise. *Proceedings of the Annual Meeting of the Cognitive Science Society* 46, 0 (2024). https://escholarship.org/uc/item/3fv0t3nq

[90] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[91] Nataasha Raul, Radha Shankarmani, and Padmaja Joshi. 2020. A Comprehensive Review of Keystroke Dynamics-Based Authentication Mechanism. In *International Conference on Innovative Computing and Communications*, Ashish Khanna, Deepak Gupta, Siddhartha Bhattacharyya, Vaclav Snasel, Jan Platos, and Aboul Ella Hassanien (Eds.). Springer, Singapore, 149–162. https://doi.org/10.1007/978-981-15-0324-5_13

[92] Kenneth Revett, Sérgio Tenreiro de Magalhães, and Henrique M. D. Santos. 2005. Enhancing Login Security Through the Use of Keystroke Input Dynamics. In *Advances in Biometrics*, David Zhang and Anil K. Jain (Eds.). Springer, Berlin, Heidelberg, 661–667. https://doi.org/10.1007/11608288_88

[93] Kenneth Revett, Florin Gorunescu, Marina Gorunescu, Marius Ene, Sergio Magalhaes, and Henrique Santos. 2007. A machine learning approach to keystroke dynamics based user authentication. *International Journal of Electronic Security and Digital Forensics* 1, 1 (Jan. 2007), 55–70. https://doi.org/10.1504/IJESDF.2007.013592 Publisher: Inderscience Publishers.

[94] 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 (UIST '20)*. Association for Computing Machinery, New York, NY, USA, 686–696. https://doi.org/10.1145/3379337.3415816

[95] Helena Roeber, John Bacus, and Carlo Tomasi. 2003. Typing in thin air: the canesta projection keyboard - a new method of interaction with electronic devices. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems (CHI EA '03)*. Association for Computing Machinery, New York, NY, USA, 712–713. https://doi.org/10.1145/765891.765944

[96] Russ Rolfe. 2007. What is an IME (Input Method Editor) and how do I use it ? (2007).

[97] John Sadowsky. 1996. Investigation of Signal Characteristics Using the Continuous Wavelet Transform. *JOHNS HOPKINS APL TECHNICAL DIGEST* 17, 3 (1996).

[98] Timothy A. Salthouse. 1984. Effects of age and skill in typing. *Journal of Experimental Psychology: General* 113, 3 (1984), 345–371. https://doi.org/10.1037/0096-3445.113.3.345 Place: US Publisher: American Psychological Association.

[99] David Slater, Scott Novotney, Jessica Moore, Sean Morgan, and Scott Tenaglia. 2019. Robust keystroke transcription from the acoustic side-channel. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC '19)*. Association for Computing Machinery, New York, NY, USA, 776–787. https://doi.org/10.1145/3359789.3359816

[100] D. Song, D. Wagner, and Xuqing Tian. 2001. Timing Analysis of Keystrokes and Timing Attacks on SSH. https://www.semanticscholar.org/paper/Timing-Analysis-of-Keystrokes-and-Timing-Attacks-on-Song-Wagner/23613cf6a3f1fbb5ec86de7f1f1ba3f535a68e21

[101] Marco Speicher, Anna Maria Feit, Pascal Ziegler, and Antonio Krüger. 2018. Selection-based Text Entry in Virtual Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174221

[102] 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 (CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1–16. https://doi.org/10.1145/3491102.3501878

[103] Zahid Syed, Sean Banerjee, Qi Cheng, and Bojan Cukic. 2011. Effects of User Habituation in Keystroke Dynamics on Password Security Policy. In *2011 IEEE 13th International Symposium on High-Assurance Systems Engineering*. 352–359. https://doi.org/10.1109/HASE.2011.16 ISSN: 1530-2059.

[104] Lynda B. Taylor, Cyril J. Weir, and Association of Language Testers in Europe (Eds.). 2008. *Multilingualism and assessment: achieving transparency, assuring quality, sustaining diversity: proceedings of the ALTE Berlin conference, May 2005* (1. publ ed.). Number 27 in Studies in language testing. Cambridge Univ. Press, Cambridge.

[105] Martin Vuagnoux and S. Pasini. 2009. Compromising Electromagnetic Emanations of Wired and Wireless Keyboards. https://www.semanticscholar.org/paper/Compromising-Electromagnetic-Emanations-of-Wired-Vuagnoux-Pasini/fcede4144d229e7474e72b3c37192b7525570ada

[106] Ahmed Wahab, Daqing Hou, Stephanie Schuckers, and Abbie Barbir. 2021. Utilizing Keystroke Dynamics as Additional Security Measure to Protect Account Recovery Mechanism:. In *Proceedings of the 7th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, Online Streaming, — Select a Country —, 33–42. https://doi.org/10.5220/0010191200330042

[107] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. 2015. MoLe: Motion Leaks through Smartwatch Sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*. Association for Computing Machinery, New York, NY, USA, 155–166. https://doi.org/10.1145/2789168.2790121

[108] Jian Wang, Rukhsana Ruby, Lu Wang, and Kaishun Wu. 2016. Accurate Combined Keystrokes Detection Using Acoustic Signals. In *2016 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*. 9–14. https://doi.org/10.1109/MSN.2016.010

[109] Junjue Wang, Kaichen Zhao, Xinyu Zhang, and Chunyi Peng. 2014. Ubiquitous Keyboard for Small Mobile Devices: Harnessing Multipath Fading for Fine-Grained Keystroke Localization. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '14)*. Association for Computing Machinery, New York, NY, USA, 14–27. https://doi.org/10.1145/2594368.2594384

[110] Junjue Wang, Kaichen Zhao, Xinyu Zhang, and Chunyi Peng. 2014. Ubiquitous keyboard for small mobile devices: harnessing multipath fading for fine-grained keystroke localization. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services (MobiSys '14)*. Association for Computing Machinery, New York, NY, USA, 14–27. https://doi.org/10.1145/2594368.2594384

[111] Xueyi Wang, Yifan Liu, and Shancang Li. 2023. Deep Learning Enabled Keystroke Eavesdropping Attack Over Videoconferencing Platforms. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 1–2. https://doi.org/10.1109/INFOCOMWKSHPS57453.2023.10225861 ISSN: 2833-0587.

[112] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. https://doi.org/10.48550/arXiv.1910.03771 arXiv:1910.03771 [cs].

[113] Wenge Xu, Hai-Ning Liang, Anqi He, and Zifan Wang. 2019. Pointing and Selection Methods for Text Entry in Augmented Reality Head Mounted Displays. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 279–288. https://doi.org/10.1109/ISMAR.2019.00026 ISSN: 1554-7868.

[114] Edwin Yang, Song Fang, Ian Markwood, Yao Liu, Shangqing Zhao, Zhuo Lu, and Haojin Zhu. 2022. Wireless Training-Free Keystroke Inference Attack and Defense. *IEEE/ACM Transactions on Networking* 30, 4 (Aug. 2022), 1733–1748. https://doi.org/10.1109/TNET.2022.3147721 Conference Name: IEEE/ACM Transactions on Networking.

[115] Qiang Yang, Yongpan Zou, Meng Zhao, Jiawei Lin, and Kaishun Wu. 2018. ArmIn: Explore the Feasibility of Designing a Text-entry Application Using EMG Signals. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous '18)*. Association for Computing Machinery, New York, NY, USA, 117–126. https://doi.org/10.1145/3286978.3287030

[116] Shanhe Yi, Zhengrui Qin, Ed Novak, Yafeng Yin, and Qun Li. 2016. GlassGesture: Exploring head gesture interface of smart glasses. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2016.7524542

[117] Xin Yi, Chun Yu, Weinan Shi, and Yuanchun Shi. 2017. Is it too small?: Investigating the performances and preferences of users when typing on tiny QWERTY keyboards. *International Journal of Human-Computer Studies* 106 (Oct. 2017), 44–62. https://doi.org/10.1016/j.ijhcs.2017.05.001

[118] Caglar Yildirim. 2023. Point and Select: Effects of Multimodal Feedback on Text Entry Performance in Virtual Reality. *International Journal of Human–Computer Interaction* 39, 19 (Nov. 2023), 3815–3829. https://doi.org/10.1080/10447318.2022.2107330 Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10447318.2022.2107330.

[119] Yafeng Yin, Qun Li, Lei Xie, Shanhe Yi, Edmund Novak, and Sanglu Lu. 2016. CamK: A Camera-Based Keyboard for Small Mobile Devices. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2016.7524400

[120] Yafeng Yin, Qun Li, Lei Xie, Shanhe Yi, Ed Novak, and Sanglu Lu. 2018. CamK: Camera-Based Keystroke Detection and Localization for Small Mobile Devices. *IEEE Transactions on Mobile Computing* 17, 10 (Oct. 2018), 2236–2251. https://doi.org/10.1109/TMC.2018.2798635 Conference Name: IEEE Transactions on Mobile Computing.

[121] Lishuang Zhan, Tianyang Xiong, Hongwei Zhang, Shihui Guo, Xiaowei Chen, Jiangtao Gong, Juncong Lin, and Yipeng Qin. 2024. TouchEditor: Interaction Design and Evaluation of a Flexible Touchpad for Text Editing of Head-Mounted Displays in Speech-unfriendly Environments. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 4 (Jan. 2024), 198:1–198:29. https://doi.org/10.1145/3631454

[122] Kehuan Zhang and XiaoFeng Wang. 2009. Peeping Tom in the Neighborhood: Keystroke Eavesdropping on Multi-User Systems. In *Proceedings of the 18th Conference on USENIX Security Symposium (SSYM'09)*. USENIX Association, USA, 17–32.

[123] 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 (CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1–16. https://doi.org/10.1145/3491102.3517686

[124] Yanchao Zhao, Yiming Zhao, Si Li, Hao Han, and Lei Xie. 2023. UltraSnoop: Placement-agnostic Keystroke Snooping via Smartphone-based Ultrasonic Sonar. *ACM Trans. Internet Things* 4, 4 (Nov. 2023), 22:1–22:24. https://doi.org/10.1145/3614440

[125] 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 (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174013