AcousticType: Smartwatch-Enabled Cross-Device Text Entry Method Using Keyboard Acoustics

ÜLKÜ METERIZ-YILDIRAN, University of Central Florida, USA
NECIP FAZIL YILDIRAN, University of Central Florida, USA
DAVID MOHAISEN, University of Central Florida, USA

The integration of various technologies with our daily lives, such as smartwatches, smartphones, smart TVs, etc. has proven beneficial. However, the HCI community is still in search for more intuitive text entry methods for smart devices. In this study, we explore a new text entry method—AcousticType for smart devices where a smartwatch is leveraged to infer what a user is typing on a physical keyboard while wearing the smartwatch employing the acoustic signals as a secondary input mechanism. To address various environment-related challenges that prevent the direct use of the acoustic signals for inferring typed keys, AcousticType employs four modules: *Noise Cancelling*, *Keystroke Detection*, *Key Identification* and *Word Correction*, where several digital signal processing, machine learning, and natural language processing techniques are utilized to produce the final inference. Our results show that an acoustic emanation of a physical keyboard captured by a smartwatch recovers up to 98% of the typed text. We also showed that utilizing the noise cancellation, AcousticType is robust to the changes in the environment, which further boosts the practicality of AcousticType. The findings are promising and call for further investigation into new types of text entry methods that utilize acoustic signals to cope with the usability issues in smart devices.

ACM Reference Format:
Ülkü Meteriz-Yıldıran, Necip Fazıl Yıldıran, and David Mohaisen. 2022. AcousticType: Smartwatch-Enabled Cross-Device Text Entry Method Using Keyboard Acoustics. 1, 1 (March 2022), 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Due to the easy access and all the conveniences the new technologies offer, our daily lives are getting more and more entwined with smart devices. According to an annual Internet report by Cisco [2], Machine to Machine (M2M) connections, which are mostly utilized by smart devices, are the fastest-growing device and connection category. Cisco also reports that smartphones, Smart TVs, media streaming devices (Fire TV, Chromecast, etc.) are the next growing categories after M2M. Despite the growing interest in smart devices, more intuitive text entry methods are still an open problem. Most smart TVs and media streaming devices are still utilizing QWERTY keyboards that are navigated through the remote controller. Text entry in smartwatches is also challenging due to the limited interaction space. This problem is also prevalent in smart glasses, such as Magic Leap 1, Microsoft Hololens, and Vuzix Blade.

Smartwatches have skyrocketed in popularity for their mobility features, which brings a lot of convenience. According to a recent market report [1], the year-over-year growth of the smartwatch market is expected to be 14.5% for 2020—2025

Authors' addresses: Ülkü Meteriz-Yıldıran, University of Central Florida, Orlando, USA, meteriz@knights.ucf.edu; Necip Fazıl Yıldıran, University of Central Florida, Orlando, USA, yildiran@knights.ucf.edu; David Mohaisen, University of Central Florida, Orlando, USA, mohaisen@ucf.edu.

and IoT-driven smartwatches are a key trend. Another study [22] suggests that smartwatches are used more frequently than smartphones nowadays. To keep up with the ever-growing user expectations, smartwatches are equipped with different I/O mechanisms, e.g., motion sensors, touch screen, heart rate sensor, thermometer, microphone, speaker, etc. Such wide range of I/O mechanisms pave the way for utilizing smartwatches for cross-device applications [5, 10].

In this work, we demonstrate a proof-of-concept to utilize smartwatches to facilitate new *cross-device text input* methods. Figure 1 shows an example use case where the user types on a keyboard to input text to smart TV which typically allows entering text only via remote controller. The biggest advantage of our system is that the keyboard is not required to be connected/linked/paired with the smart TV in any way for our input method. Such use cases can be elaborated with numerous smart devices that can be connected to smartwatch via Bluetooth, WiFi, etc. or even to augment the input capabilities of the smartwatch itself. For example, the input capabilities of smartwatches are quite limited due to their small screen sizes, and our method offers a convenient option for entering text on smartwatches through a secondary source. Stand-alone smartwatches have Internet connection capabilities, thus users can utilize them for e-mailing and instant messaging (IM). With our method, a user can compose an e-mail or a quick IM using a keyboard that is not connected/linked/paired to the smartwatch in any way. Another use case could be text entry on augmented reality (AR) head mounted displays (HMD), where the current text entry methods are limited in terms of usability. For instance, most of the AR HMD keyboards are operated by remote controller, head movements + gestures, gaze + gestures, and hand tracking + gestures. Although the usability scores are improving with the new advancements, they are still not close to that of the conventional physical keyboards. Our method can be utilized for AR HMDs to facilitate input entry. For example, users can start a typing session on their smartwatch and use their wired/wireless keyboard connected to an office computer to quickly reply to a personal e-mail on an AR HMD.

Our aim is AcousticType, an inference framework utilizing keyboard acoustic emanations captured by a smartwatch microphone. The goal is to recover characters typed by a user using the keyboard acoustic emanations captured by the user's smartwatch. We develop and demonstrate a smartwatch application that offers seamless cross-device interaction as illustrated in Figure 2. The application uses user-tailored inference models or generic inference models for each physical keyboard model. Users select the physical keyboard model and the target smart device in the setup stage, and start typing on the keyboard. Such application utilizing AcousticType is capable of converting conventional wired, wireless or legacy keyboards to Bluetooth keyboards through smartwatches.

Contributions. In this study, we make the following unique contributions. (1) We propose an inference framework, AcousticType, that can be utilized in a smartwatch application as a secondary text input mechanism. AcousticType identifies keystrokes using the acoustic emanations captured by smartwatches. (2) We highlight that in a *trusted setting*, AcousticType enables new text entry mechanisms for smart devices. (3) We build a four-stage pipeline: *Noise Cancelling*, *Keystroke Detection*, *Key Identification* and *Word Correction*. The noise cancelling stage allows us to neutralize the environmental changes that is never considered in previous studies. With the representative audio features and learning techniques, we identify the keystrokes. On top of the baseline performance, which is already promising, we employ state-of-the-art natural language processing techniques to improve AcousticType's accuracy by word correction. (4) We present an extensive set of experiments showing that AcousticType can recover up to 98% of the text. (5) We collected three types of test datasets to examine how our inference framework performs under different types of use patterns.

Reproducibility. All artifacts associated with our work are available at our Git Repository.

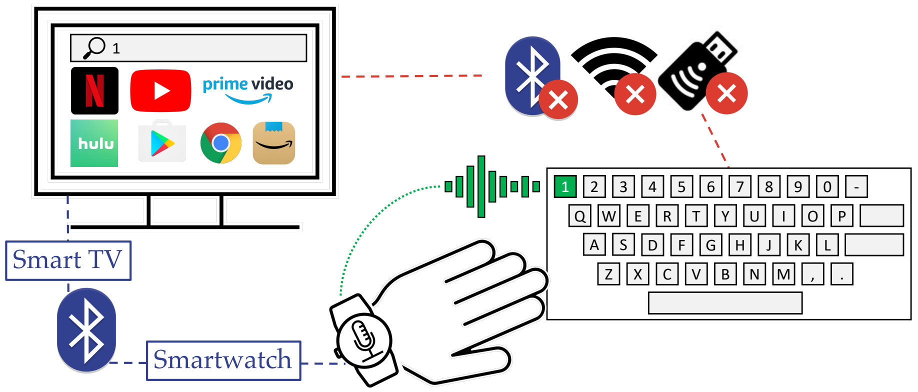


Fig. 1. An example use case. User starts a typing session on smartwatch that is connected to smart TV. Then, user types on any keyboard that is not capable of connecting the smart TV. The acoustic emanations are processed by the smartwatch running AcousticType and the text input feed are forwarded to smart TV.

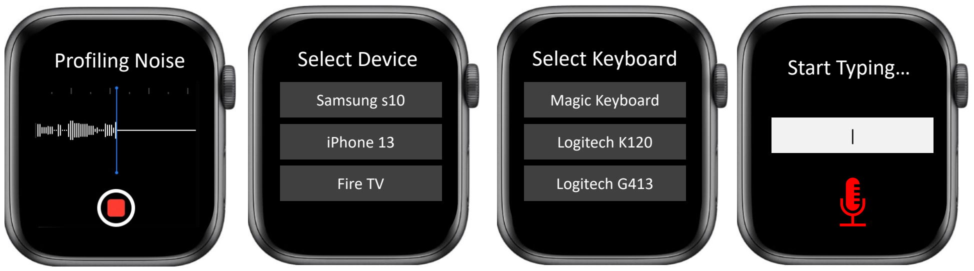


Fig. 2. An example GUI guiding through the application setup. First, the application performs noise profiling for the current typing session. Then, the user selects the device which the keyboard inputs will be forwarded. Next, the user selects the keyboard model. Once the application prepares the associated model, it starts listening for keystroke events.

2 METHODOLOGY

Our inference framework has four stages as illustrated in Figure 3: *Noise Cancelling*, *Keystroke Detection*, *Key Identification*, and *Word Correction*. The framework takes the raw signal as input and outputs the final prediction.

2.1 Noise Cancelling

Since the recording devices record every sound that exists in the environment, they record the background noises. The alternations of the background noise significantly affect the performance of the identification. Therefore, as a first step, we prepare the data by cleaning any background noise. Two types of background noise exist: (i) the white noise and (ii) other ambient noise in the environment, e.g., street noises, computer fan noise, etc. For both types of background noises, our noise cancelling algorithm utilizes the Fourier analysis where the fingerprint of the static background noise is structured using the spectrum of the pure tones in the quiet parts of the recording. The fingerprinting is a crucial step, since cleaning the frequency bands of the noise directly from the recording also cleans the actual keystroke acoustics. After fingerprinting the noise, it is cancelled from the raw signal and the clean signal is forwarded to *Keystroke Detection*.

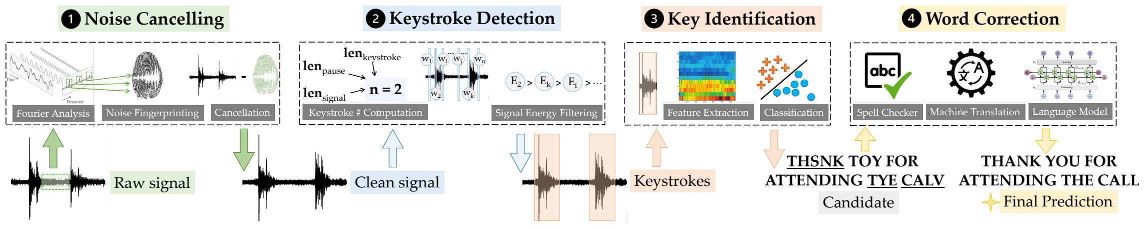


Fig. 3. AcousticType consists of four main stages. *Noise Cancelling* takes the raw signal, cancels any ambient background noise or white noise (hiss) from the signal, and returns the clean signal. *Keystroke Detection* takes the clean signal and returns 200 ms windows encapsulating the keystroke events. *Key Identification* takes the windows and predicts the associated characters. *Word Correction* takes the predictions and corrects the misspelled words.

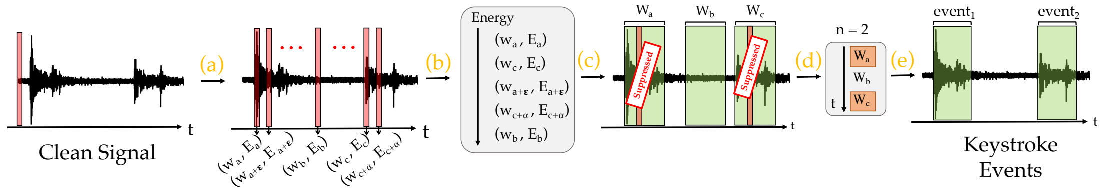


Fig. 4. *Keystroke detection* pipeline. **(a)** A 10 ms window is slid over the clean signal. The spectrum energy, E_a, of each 10 ms windows w_a is calculated. **(b)** The window-energy tuples, (w_a, E_a), are sorted in terms of the energy. **(c)** Starting from the most powerful 10 ms window, 200 ms windows, W_a, encapsulating keystroke events are created. Whenever a new window is created, the subsequent 10 ms windows $(w_{a+\epsilon})$, which are less powerful and overlaps with it, are suppressed. **(d)** The maximum number of keystrokes, n, is computed considering the length of the clean signal coming from *Noise Cancelling* stage. The n-most powerful 200 ms windows are fetched and sorted in time. **(e)** *Keystroke Detection* returns a set of windows encapsulating the keystroke events.

In practice, to fingerprint the noise profile, the smartwatch will ask the user to stay silent for a few seconds and analyze the background noises to create the fingerprint as illustrated in Figure 2. For the smartwatches utilizing the emergent hardware-level noise cancelling technologies, such as the microphone in the latest-generation Apple Watch, this stage can be skipped to further improve the responsiveness by reducing the latency.

2.2 Keystroke Detection

In this stage, we locate the keystroke events in the clean signal based on two observations: (i) a keystroke event lasts typically 200 ms and yields two peaks, a hit peak and a release peak; (ii) the hit peak is more powerful than the release peak. Thus, we simplify the keystroke detection problem to locating the hit peak in a keystroke event which typically lasts 10 ms. As illustrated in Figure 4, we slide a 10 ms window, over the clean signal and calculate the spectrum energy along the way. As we expect a high energy in case of a hit peak, we sort the 10 ms windows according to their energies and encapsulate them with 200 ms windows. Then, we calculate the maximum number of keystrokes, n, considering the typical duration of a keystroke (200 ms), the pause (500 ms) between keystrokes, and the total length of the recording. Finally, we return the top n *non-overlapping* 200 ms windows as the keystroke events.

As we do not have access to the live stream of acoustics from smartwatch in this proof-of-concept work, we performed *Keystroke Detection* offline. However, this task can be performed in real-time, once a typing session is initiated on the smartwatch application. In a typing session, the application will be listening for keystroke events and will perform *Keystroke Detection* every 500 ms—which contains at most two keystrokes as illustrated in Figure 4.

2.3 Key Identification

This stage consists of two parts: (i) feature extraction and (ii) classification.

To decide the feature extraction method, we experimented with the most successful audio features and compared the average true positive rates (TPR) they produce. We considered Fast Fourier Transform (FFT) coefficients (0.07), Cepstrum coefficients (0.7), Mel-Frequency Cepstral Coefficients (MFCC) (0.85), and Chroma (0.37) features. Based on the empirical results, we decided to use MFCC as our audio features.

Similarly, for the classification, we experimented on various classification techniques and compared their TPR. We considered logistic regression (0.83), support vector machines (0.85), multi-layer perceptron (0.73), and convolutional neural-network (0.73). Based on the results, we decided to use support vector machine, which is also favorable as SVM yields a lightweight model that can be easily used in smartwatches or smartphones that smartwatches are paired with.

2.4 Word Correction

Key Identification stage may produce misspelled word predictions due to the signal similarities, especially between the keys in close proximity. To further improve the prediction, we added *Word Correction* stage where we evaluated three methods: Simple Spell Checker, Machine Translation, and Next Word Prediction.

In Simple Spell Checker (SSC), an algorithm based on the Levenshtein distance is used to find permutations within an edit distance of 2 from the original word [17]. Levenshtein distance [13] is a string metric that measures the distance between two strings. It counts the minimum number of single-character edits, such as insertion, deletion, or substitution required to make two strings identical. After finding all candidates within Levenshtein distance of 2, all permutations are compared to known words in a word frequency list. Those words that are found more often in the frequency list are more likely the correct results.

Machine Translation (MT) utilizes one of the most advanced approaches in NLP: Transformers. With transformers, we convert the spell correction task to machine translation task. We consider the misspelled words and the correct words as the associated translation and trained the network. Then, we use the model to correct the misspelled words. For MT, we used xfspell tool developed by Hagiwara.

In *Key Identification*, we observed that some predictions are not misspelled but do not align with the context of the sentence. For such cases, we use Next Word Prediction (NW) to predict the next word given the context of a sentence. NW utilizes state-of-the-art GPT-2 [19] language model to replace the out-of-context words. We feed the word in the sentence sequentially and get the next word predictions. Then, we replace the original word and compare the sentence score. If the score does not improve, we keep the original word.

We note that, as smartphones have user-tailored language models for text entries, the smartwatch application that is paired with a smartphone can directly use that language model to correct words. Such specifically trained language models would further improve the performance.

3 EVALUATION AND DISCUSSION

In this section, we outline the data collection process and discuss our findings from two sets of evaluations: User Profiling and Generic Profiling.

3.1 Data Collection

For evaluations, we collected training and testing data using Samsung Galaxy Watch Active 2 smartwatch as the recording device and MacBook Air 2020 (MBA) and Magic Keyboard (MK) as the target keyboards. We collected data from two users where they wore the watch and positioned the keyboard however they felt comfortable. For training data, first user typed 45 pangrams to train the key identification model. For testing data, we collected three types of data from both users: (i) a randomly selected e-mail from Enron e-mail dataset [12] (**E-mail**), (ii) 32 randomly generated passwords of length 8 (**Random**), (iii) 20 randomly selected passwords from RockYou password dataset (**Selected**).

3.2 Evaluation Metrics

For our experiments, we used four metrics: (i) cross-entropy, (ii) character-wise true positive rate (TPR), (iii) string-wise true positive rate (TPR), and (iv) normalized Levenshtein distance.

Cross-Entropy. Cross-entropy measures the difference between two probability distributions for a given random variable. To evaluate the performance of our classifier, we used the cross-entropy of the predicted probability distribution, f, relative to the actual distribution, p. The cross-entropy is calculated as follows:

$$H(p, f) = -\sum_{i} p(x_i)\log(f(x_i) + \epsilon) \tag{1}$$

The ϵ value in (1) is used to avoid the undefined values yielded by log(0). For our evaluations, we calculated (i) the cross-entropy between the probability distribution that our model estimates and that of the actual distribution, $H(p, f_{\text{SVM}})$, (ii) the cross-entropy between the character distribution of the associated dataset and the actual distribution, $H(p, f_u)$, and (iii) the cross-entropy between the actual distribution and itself representing the ideal cross-entropy, $H(p, p)$. $H(p, p)$ is calculated as a reference point for all other cross-entropy calculations, which is negligible.

Character-wise TPR. As each key on the keyboard represents a class in the classifier, we use character-wise TPR to measure the performance of *Key Identification* stage. We report the TPR for each alphanumeric key on the keyboard.

String-wise TPR. We utilize string-wise TPR to compare the predicted strings with their associated ground truths.

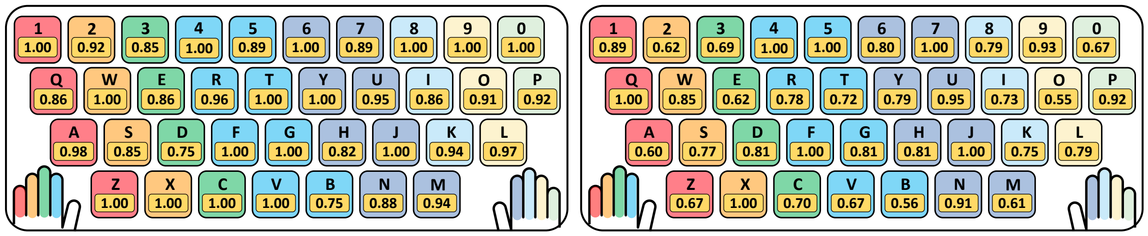
Normalized Levenshtein Distance (NLD). As briefly mentioned above, LD is a string metric that measures the difference between two strings. It measures the minimum number of edits (insertion, deletion, or substitution) to make the strings identical. For our experiments, we used its normalized variant as a string comparison metric. NLD is calculated by dividing the LD between two strings (predicted and target) by the length of the target string.

For string-wise TPR and NLD, we also consider their one-hop variations due to the fact that some of the mispredicted letters are within the one-hop distance from the actual key. This is reasonable as the acoustics from the keys that are close are similar to one another.

3.3 User Profiling

In this set of experiments, we used the training and testing data collected from the first user. This evaluation assumes that the user will train the model to capture the unique typing style.

Character-wise Evaluations. Figure 5 shows the character-wise TPR for each keyboard. The TPR values range from 0.75 to as high as 1.00, which demonstrates the success of AcousticType on the MBA keyboard. Figure 5a also shows the location of each key and the colors emphasize the finger that presses that key. The most successful predictions, with an



1 1.00, 2 0.92, 3 0.85, 4 1.00, 5 0.89, 6 1.00, 7 0.89, 8 1.00, 9 1.00, 0 1.00
Q 0.86, W 1.00, E 0.86, R 0.96, T 1.00, Y 1.00, U 0.95, I 0.86, O 0.91, P 0.92
A 0.98, S 0.85, D 0.75, F 1.00, G 1.00, H 0.82, J 1.00, K 0.94, L 0.97
Z 1.00, X 1.00, C 1.00, V 1.00, B 0.75, N 0.88, M 0.94
1 0.89, 2 0.62, 3 0.69, 4 1.00, 5 1.00, 6 0.80, 7 1.00, 8 0.79, 9 0.93, 0 0.67
Q 1.00, W 0.85, E 0.62, R 0.78, T 0.72, Y 0.79, U 0.95, I 0.73, O 0.55, P 0.92
A 0.60, S 0.77, D 0.81, F 1.00, G 0.81, H 0.81, J 1.00, K 0.75, L 0.79
Z 0.67, X 1.00, C 0.70, V 0.67, B 0.56, N 0.91, M 0.61

(a) MacBook Air (b) Magic Keyboard

Fig. 5. Character-wise TPR obtained in *Key Identification* stage for user profiling. The key colors encode the hand and finger used.

Table 1. User profiling results for *Key Identification* and *Word Correction*. *Word Correction* results are written in *italic*. *Word Correction* is only applicable for E-mail dataset as other datasets do not include English words. TPR^1 and NLD^1 stand for one-hop variations of string-wise TPR and NLD.

	MacBook Air				**Magic Keyboard**			
	TPR	**TPR^1**	**NLD**	**NLD^1**	**TPR**	**TPR^1**	**NLD**	**NLD^1**
E-mail	0.918	0.988	0.065	0.009	0.796	0.918	0.164	0.065
SSC	*0.970*	*0.988*	*0.023*	*0.009*	*0.877*	*0.930*	*0.023*	*0.009*
MT	*0.953*	*0.994*	*0.037*	*0.004*	*0.970*	*0.988*	*0.023*	*0.009*
NW	*0.866*	*0.936*	*0.112*	*0.053*	*0.872*	*0.953*	*0.103*	*0.037*
Random	0.972	0.992	0.025	0.007	0.812	0.921	0.177	0.073
Selected	0.878	0.939	0.110	0.055	0.679	0.834	0.290	0.150

average of 0.96, are done on the keys pressed with the left little finger, left index finger, right ring finger, and right little finger. The least successful predictions are recorded with the left middle finger with an average of 0.86.

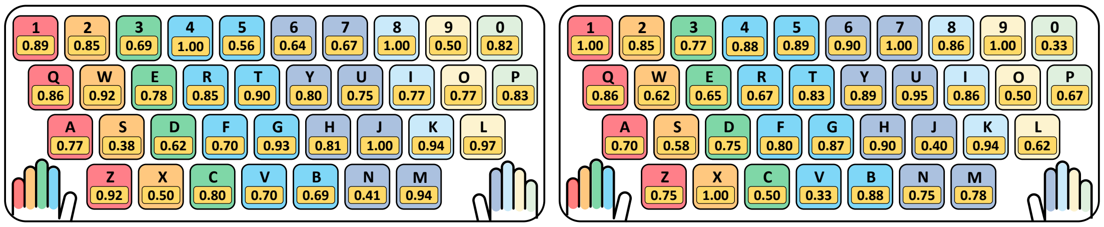
Figure 5b shows the character-wise TPR for the MK keyboard. The results range from 0.55 to 1.00. For the MK keyboard, the highest TPR, an average of 0.85, was observed with the left index finger. The smallest TPR is observed with the right middle finger, with an average of 0.70. When the used hands are considered, the TPR for the right hand (MBA $\longrightarrow$ 0.94, MK $\longrightarrow$ 0.81) is slightly higher than that of the left hand (MBA $\longrightarrow$ 0.93, MK $\longrightarrow$ 0.78). This behavior is reasonable, because the smartwatch is worn to the left wrist, and the microphone stays still when the right hand is used.

String-wise Evaluations. Table 1 shows the user profiling results. As word correction is language-specific (i.e., English in our case), it is only evaluated for the E-mail dataset. With user profiling, we can infer 97% of the typed text. Considering one-hop variations, TPR goes up to 99%. The average increase of 10% with the one-hop variations supports our claim: some of the mispredicted characters are within the one-hop distance from the actual keys on the keyboard.

When we consider *Word Correction* results, SSC and MT improved TPR in all cases. However, NW is shown to decrease the TPR in some cases. The reason is that NW may replace the misspelled word (e.g., "nusiness" for "business") with another word (e.g., "work") fitting in the sentence context. Such replacement may decrease the TPR in some cases, leaving some room for improvements.

3.4 Generic Profiling

In this set of experiments, we used the training data collected from the first user to train our model and the testing data from the second user for the evaluations. Unlike user profiling, we use a generic model to infer the typed text.



(a) MacBook Air (b) Magic Keyboard

Fig. 6. Character-wise TPR obtained in *Key Identification* stage for generic profiling.

Table 2. Generic profiling results for *Key Identification* and *Word Correction*. *Word Correction* results are written in *italic*. *Word Correction* is only applicable for E-mail dataset as other datasets do not include English words. TPR^1 and NLD^1 stand for one-hop variations of string-wise TPR and NLD.

	MacBook Air				**Magic Keyboard**			
	TPR	**TPR^1**	**NLD**	**NLD^1**	**TPR**	**TPR^1**	**NLD**	**NLD^1**
E-mail	0.680	0.877	0.258	0.098	0.761	0.930	0.192	0.056
SSC	*0.790*	*0.918*	*0.169*	*0.066*	*0.848*	*0.959*	*0.122*	*0.032*
MT	*0.709*	*0.883*	*0.234*	*0.093*	*0.709*	*0.883*	*0.234*	*0.093*
NW	*0.734*	*0.878*	*0.214*	*0.098*	*0.734*	*0.878*	*0.214*	*0.098*
Random	0.792	0.949	0.195	0.047	0.769	0.921	0.217	0.073
Selected	0.758	0.912	0.210	0.070	0.762	0.933	0.215	0.060

Table 3. Cross-entropy results for User Profiling and Generic Profiling.

	Baseline		User Profiling		Generic Profiling	
	Ideal	**Frequency**	**MacBook Air**	**Magic Keyboard**	**MacBook Air**	**Magic Keyboard**
E-mail	ϵ	14.02	0.66	0.65	1.19	1.29
Random	ϵ	3.58	0.40	0.33	1.29	1.12
Selected	ϵ	18.98	0.86	0.80	1.09	1.18

Figure 6 demonstrates the character-wise TPR for generic profiling. When we compare the results in Figure 5 and Figure 6, we observe a slight decrease in the overall TPRs for the MK (0.798 $\longrightarrow$ 0.764) and a relatively large decrease for the MBA (0.937 $\longrightarrow$ 0.776). The decrease is expected since User Profiling tailors the classifier to capture unique typing style of the user. On the other hand, Generic Profiling only models the acoustic emanations of the keyboard.

Table 2 shows the results for the generic profiling. With the generic profiling, we are able to infer up to ~85% of the typed text. When we compare Table 1 and Table 2, we observe a decrease in the prediction in TPR. However, the rate of decrease is smaller when we consider the one-hop variations. This shows that most of the mispredicted characters are within the one-hop distance of the correct character. This observation gives some hints about the mispredicted characters, which further reduces the search space for the corrected version of the typed text.

3.5 Cross-Entropy

To evaluate how much our inference reduces the entropy, we computed the cross-entropy between the actual probability distribution, which is a one-hot vector (1 on the correct class, 0 for others), and the probability distribution returned by

Table 4. A summary of the related work. The detection is measured by TPR (True Positive Rate). UP in training column stands for User Profiling, and HM in the exploited medium column stands for Hand Movement. Blank proximity and detection indicate that those values are not provided in the corresponding study.

Reference	**Year**	**Exploited Medium**	**Proximity**	**Training?**	**Performance**
Asonov et al. [3]	2004	Keyboard acoustics	1m	Yes (UP)	79% (top-1 key)
Berger et al. [4]	2006	Differences of keyboard acoustics	-	No	73% (top-50 word)
Zhuang et al. [25]	2009	Bootstrapped keyboard acoustics	-	No	90% (top-1 word)
Halevi et al. [7]	2014	Keyboard acoustics	-	Yes	64% (top-1 key)
Wang et al. [24]	2015	Dislocation of hand	Smartwatch	Yes	30% (top-5 word)
Liu et al. [14]	2015	HM+Acoustic emanations	Smartwatch	Yes	55% (top-5 word)
Maiti et al. [16]	2016	HM+Acoustic emanations	Smartwatch	Yes	51% (top-10 word)
Wang et al. [23]	2016	Hand movement	Smartwatch	No	80% (top-1 PIN)
Compagno et al. [6]	2016	Acoustics via VoIP	Remote	Yes (UP)	83% (top-1 key)
AcousticType	2021	Acoustics via Smartwatch	Smartwatch	Yes (UP)	98% (top-1 key)
AcousticType	2021	Acoustics via Smartwatch	Smartwatch	Yes	85% (top-1 key)

the classifier. Table 3 shows the cross-entropy results for both user profiling and generic profiling. The perfect match with the actual distribution ("Ideal" column in Table 3) is a negligible value ϵ. The **Frequency** column in Table 3 shows different cross-entropy values for the different datasets considering the character frequency for the corresponding domain. For the E-mail dataset, **Frequency** column shows the cross-entropy with the character probability distribution of the English language. For the "Selected" dataset, it shows the cross-entropy with the character probability distribution against the RockYou leaked password dataset. For the "Random" dataset, it shows the cross-entropy against the uniform distribution. The entropy loss introduced for both UP and GP demonstrates the effectiveness of our inference mechanism.

4 RELATED WORK

In this section, we discuss the literature that is related to AcousticType across various dimensions.

Keylogging. In away, and despite the split environment as a result of the envisioned application of AcousticType where the input is seen on a keyboard and the effect is realized on another separated device through inference, our work exhibits some similarities with the literature of keylogging. The previous keylogging studies draw attention to the adversarial usage of acoustics and smartwatches. Although the context differs, the end goal of all studies is the same: inferring what a user is typing on a keyboard.

Table 4 compares AcousticType with other works. When we compare the studies utilizing acoustics [3, 4, 6, 7, 25], our inference pipeline outperforms other methods. However, we should note that the evaluation methods employed in these studies do not consider the effect of environmental changes, i.e., background noises, which we address.

Considering the studies that utilizes smartwatches but different features [14, 16, 23, 24], we note they use different evaluation metrics. For instance, the top-x word means that the study matches words instead of keys, since they are matching the hand movement patterns with the corresponding words. The top-x PIN means that the study matches only numerical keys. When we assume that an average word length is 5, our top-1 word accuracy becomes 90% and our method again outperforms their identification accuracy.

Smartwatch-Enabled Cross-Device Interactions. The cross-device interaction strategies are explored for wide range of devices. Houben and Marquardt [10] developed a toolkit utilizing smartwatches for cross-device interaction. The toolkit offer various interaction methods with smartwatches such as *on the watch, above the watch*, etc. to be used in cross-device interactions. Chen *et al.* [5] introduced *Duet* where they leveraged the smartwatches to capture certain gestures which further improve the interaction space on smartphones. Schmidth *et al.* [20] proposed PhoneTouch where users can use their phones to interact with surfaces. In Stitching, Hinkley *et al.* [8] developed a new interaction method that combines pen-enabled devices with wireless networking through pen gestures.

Usable Smartwatch Keyboard Designs. The research community has spent extensive efforts on usable keyboard designs for smartwatches. Although we briefly discuss selected previous works in this domain, we refer the reader to an extensive literature review by Luna *et al.* [15] who comprehensively study the text entry methods on smartwatches.

Jang *et al.* [11] developed a one-page smartwatch keyboard that is optimized for rectangle smartwatches. It uses only one page to minimize navigation while typing on a smartwatch. They reduce the 26 alphabetic keys to 13 large keys with dual input. With ZoomBoard, Oney *et al.* [18] introduced a QWERTY keyboard design for small devices including smartwatches. They implemented iterative zooming to enlarge keys on the keyboard for comfortable and accurate typing. Hong *et al.* [9] solved the fat fingers problem with SplitBoard, which splits the keyboard on watch-sized screens. VelociTap [21] employs a probabilistic model and sentence-based encoding to decide which key is pressed.

5 LIMITATIONS AND FUTURE WORK

Despite the very promising results in this work, which show the successful execution of AcousticType, this research direction is in its infancy, and is subject to several limitations. Some of those limitations are a direct result of constraints we had while executing this work, while others are general, and are worth further considerations by the community. In the following, we outline those limitations and associated future works for completeness.

First, due to COVID-19 restrictions, this study had to be performed on a limited number of test subjects. In the future, we are going to perform the experiments on multiple subjects to show the applicability at scale—guided by the robustness of the features employed, and the subject generalization results provided in this study, we expect that the results will hold on a large number of subjects. Second, although we reported the performance across different widely used keyboards, the variety of the recording devices and keyboard models are limited. In our future work, we will utilize different smartwatch and keyboard models, specifically the ones that are widely used. Third, similar to the previous studies, the data is collected in a relatively quiet environment (i.e., in a room with a wall adjacent to a public street), when compared to a recording while actually on the street, or at a café. Therefore, the effect of excessive background noise is not considered in this study, which will be explored in the future. Finally, in our experiments we performed the inferences offline as a *proof-of-concept*. While those results are very promising, and demonstrate the technical insight and practical potential of the ideas employed for coming up with AcousticType, they are limited by the narrow deployment. In our future work, we will develop the introduced smartwatch application which internally utilizes AcousticType and measure its usability in the wild with metrics other than accuracy, e.g., across response time.

6 CONCLUSION

In this work, we demonstrate a proof-of-concept for a new input mechanism for smart devices. For our inference framework, AcousticType, we leverage the acoustic emanations of conventional physical keyboards captured by a smartwatch. By neutralizing the environmental changes, we lay a foundation for a robust identification framework which address challenges unique to the environment, are never considered before. We performed two types of experiments on popular devices: user profiling and generic profiling, where we recover up to 98% and 85% of the typed text, respectively. We believe the high accuracy and the practicality of AcousticType, using quite challenging settings, will pave the way for investigation of new text entry methods for smart devices.

REFERENCES

[1] 2020. Smartwatch Market - Growth, Trends, Forecasts (2020 - 2025). https://www.researchandmarkets.com/reports/4591978/smartwatch-market-growth-trends-forecasts

[2] 2021. Cisco Annual Internet Report (2018–2023). https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[3] D. Asonov and R. Agrawal. 2004. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*. 3–11. https://doi.org/10.1109/SECPRI.2004.1301311

[4] Yigael Berger, Avishai Wool, and Arie Yeredor. 2006. Dictionary Attacks Using Keyboard Acoustic Emanations. In *Proceedings of the 13th ACM Conference on Computer and Communications Security* (Alexandria, Virginia, USA) *(CCS '06)*. Association for Computing Machinery, New York, NY, USA, 245–254. https://doi.org/10.1145/1180405.1180436

[5] Xiang 'Anthony' Chen, Tovi Grossman, Daniel J. Wigdor, and George Fitzmaurice. 2014. Duet: Exploring Joint Interactions on a Smart Phone and a Smart Watch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 159–168. https://doi.org/10.1145/2556288.2556955

[6] Alberto Compagno, Mauro Conti, Daniele Lain, and Gene Tsudik. 2017. Don't Skype & Type! Acoustic Eavesdropping in Voice-Over-IP. In *Proceedings ACM on Asia Conference on Computer and Communications Security* (Abu Dhabi, United Arab Emirates). ACM, New York, NY, USA, 703–715. https://doi.org/10.1145/3052973.3053005

[7] Tzipora Halevi and Nitesh Saxena. 2014. Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios. *International Journal of Information Security* 14 (09 2014), 1–14. https://doi.org/10.1007/s10207-014-0264-7

[8] Ken Hinckley, Gonzalo Ramos, Francois Guimbretiere, Patrick Baudisch, and Marc Smith. 2004. Stitching: Pen Gestures That Span Multiple Displays. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (Gallipoli, Italy) *(AVI '04)*. Association for Computing Machinery, New York, NY, USA, 23–31. https://doi.org/10.1145/989863.989866

[9] Jonggi Hong, Seongkook Heo, Poika Isokoski, and Geehyuk Lee. 2015. SplitBoard: A Simple Split Soft Keyboard for Wristwatch-Sized Touch Screens. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 1233–1236. https://doi.org/10.1145/2702123.2702273

[10] Steven Houben and Nicolai Marquardt. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 1247–1256. https://doi.org/10.1145/2702123.2702215

[11] Rhongho Jang, Changhun Jung, David Mohaisen, Kyunghee Lee, and Daehun Nyang. 2021. A One-Page Text Entry Method Optimized for Rectangle Smartwatches. *IEEE Transactions on Mobile Computing* (2021), 1–1. https://doi.org/10.1109/TMC.2021.3057226

[12] Bryan Klimt and Yiming Yang. 2004. The Enron Corpus: A New Dataset for Email Classification Research. In *Proceedings of the 15th European Conference on Machine Learning* (Pisa, Italy) *(ECML'04)*. Springer-Verlag, Berlin, Heidelberg, 217–226. https://doi.org/10.1007/978-3-540-30115-8_22

[13] Vladimir I Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10 (February 1966), 707.

[14] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. 2015. When Good Becomes Evil: Keystroke Inference with Smartwatch. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15)*. Association for Computing Machinery, New York, NY, USA, 1273–1285. https://doi.org/10.1145/2810103.2813668

[15] Mateus Machado Luna, Fabrizzio Alphonsus Alves de Melo Nunes Soares, Hugo Alexandre Dantas do Nascimento, Joyce Siqueira, Eduardo Faria de Souza, Thamer Horbylon Nascimento, and Ronaldo Martins da Costa. 2018. Text Entry on Smartwatches: A Systematic Review of Literature. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 02. 272–277. https://doi.org/10.1109/COMPSAC.2018.10242

[16] Anindya Maiti, Oscar Armbruster, Murtuza Jadliwala, and Jibo He. 2016. Smartwatch-Based Keystroke Inference Attacks and Context-Aware Protection Mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (China) *(ASIA CCS '16)*. Association for Computing Machinery, New York, NY, USA, 795–806. https://doi.org/10.1145/2897845.2897905

[17] Peter Norvig. 2007. https://norvig.com/spell-correct.html

[18] Stephen Oney, Chris Harrison, Amy Ogan, and Jason Wiese. 2013. ZoomBoard: A Diminutive Qwerty Soft Keyboard Using Iterative Zooming for Ultra-Small Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2799–2802. https://doi.org/10.1145/2470654.2481387

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

[20] Dominik Schmidt, Fadi Chehimi, Enrico Rukzio, and Hans Gellersen. 2010. PhoneTouch: A Technique for Direct Phone Interaction on Surfaces. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology* (New York, New York, USA) *(UIST '10)*. Association for Computing Machinery, New York, NY, USA, 13–16. https://doi.org/10.1145/1866029.1866034

[21] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating Fast Mobile Text Entry Using Sentence-Based Decoding of Touchscreen Keyboard Input. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 659–668. https://doi.org/10.1145/2702123.2702135

[22] Aku Visuri, Zhanna Sarsenbayeva, Niels van Berkel, Jorge Goncalves, Reza Rawassizadeh, Vassilis Kostakos, and Denzil Ferreira. 2017. *Quantifying Sources and Types of Smartwatch Usage Sessions*. Association for Computing Machinery, New York, NY, USA, 3569–3581. https://doi.org/10.1145/3025453.3025817

[23] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. 2016. Friend or Foe?: Your Wearable Devices Reveal Your Personal PIN. 189–200. https://doi.org/10.1145/2897845.2897847

[24] 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* (Paris, France) *(MobiCom '15)*. Association for Computing Machinery, New York, NY, USA, 155–166. https://doi.org/10.1145/2789168.2790121

[25] Li Zhuang, Feng Zhou, and J. D. Tygar. 2009. Keyboard Acoustic Emanations Revisited. *ACM Trans. Inf. Syst. Secur.* 13, 1, Article 3 (Nov. 2009), 26 pages. https://doi.org/10.1145/1609956.1609959