Sensors and Materials, Vol. 36, No. 10 (2024) 4519–4534
MYU Tokyo

S & M 3815

WatchLogger: Keystroke Detection and Recognition of Typed Words Using Smartwatch

Gangkai Li,* Yugo Nakamura, Hyuckjin Choi, Shogo Fukushima, and Yutaka Arakawa

ISEE, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan

(Received July 16, 2024; accepted September 17, 2024)

Keywords: smartwatch, wearable sensor, keystroke detection, word classification, side-channel attack

Nowadays, more and more people are wearing smartwatches in their daily lives. The various sensors embedded in smartwatches bring the ability to evaluate users' status as well as the risk of privacy issues. For example, if users are typing on keyboards while wearing smartwatches, the attacker can know the typed contents from the sensor data collected by the malicious applications that are installed on the targets' smartwatches. In this paper, we propose WatchLogger, a framework using audio and accelerometer signals to recognize the English words being typed, to demonstrate how to implement the smartwatch-based side-channel attack. In contrast with previous studies that focused on the recognition of each key or pair of keys being pressed, WatchLogger aims to perform recognition on the scale of words. To achieve this goal, WatchLogger exploits the audio signals for segmentation and the accelerometer signals for classification. In addition, we propose an ensemble classification model to deal with the problem caused by too many words. Finally, we build the WTW-100 dataset (Wearable Typed Words dataset with 100 classes of words) using data from four participants and conduct experiments on the basis of this dataset. The experimental results show accuracies of 98.31 and 99.62% and F1 scores of 0.9745 and 0.9855 for keystroke detection and classification, respectively, and an accuracy of 79.76% for word classification, indicating a considerable performance of WatchLogger.

1. Introduction

Smartwatches are becoming ubiquitous in people's daily lives. These portable smart devices are equipped with various sensors, such as an optical sensor, an accelerometer, and a microphone, making it possible to provide useful applications for recognizing users' status, such as heart rate monitoring, sports logging, and fall detection. However, these sensors also carry sensitive information that could be potentially invasive, especially when the devices are so compact and nonintrusive that users could easily neglect them and relax their vigilance. Attackers can collect sensor data from target users through a malicious application installed on the smartwatch and transmit the data to attackers' machines so that attackers can infer certain information about the targets, posing a threat to the privacy of smartwatch users.

*Corresponding author: e-mail: li.gangkai.476@s.kyushu-u.ac.jp
https://doi.org/10.18494/SAM5237

ISSN 0914-4935
https://myukk.org/

Assuming that a user is typing using a keyboard while wearing a smartwatch, the attacker can clearly infer the typed contents from the sensor data of smart devices.[1–3] Intuitively, the typing event consists of three types of hand activity, namely, moving toward the key, pressing, and releasing. When typing different words, the trajectories of hand movements could be different, as well as the fingers used to press keys. These subtle differences are reflected in the unique patterns of wrist motions that can be measured by a wrist-worn accelerometer. Therefore, the typed contents can be inferred from accelerometer signals. In addition, the sounds of keystrokes provide extra information, such as whether someone is typing or not, and can be obtained from the microphone embedded in the smartwatch.

Most previous works focus on the recognition of each key or pair of keys being pressed. Harrison *et al.*[4] proposed a framework that classifies 36 keystrokes on a MacBook keyboard on the basis of their sounds. Maiti *et al.*[1] examined the hand movement directions while typing and inferred the typed words from the series of directions. The advantage of these methods is that the number of target classes is small and fixed. However, the keystroke samples are hardly distinguishable owing to the short period of pressing one key (e.g., less than 1 s) and are vulnerable to external interference, such as environmental noise. Another problem is that using accelerometer signals to recognize each keystroke is nearly impossible because the trajectory of the hand when pressing one key depends not only on the current key but also on the previous key (e.g., the movements of pressing "e" in the words "are" and "be" are different), making it difficult to find a solution on the scale of a single key. In conclusion, the deficiency of key-by-key–based methods is that the information contained in pressing a single key is limited, making the system less robust.

In this paper, instead of recognizing each key, we propose WatchLogger, a method that recognizes the scale of words using both audio and accelerometer signals. To enable word recognition, we propose three assumptions. First, only English words will be recognized. Second, all words in a sentence are segmented by the space key. Third, we only focus on the keys of lower-case a–z and the space key. Under these assumptions, WatchLogger is able to segment the signals into frames that represent each single word by finding all space keystrokes and can train a word classifier using the labeled frames. One problem is how to find space keystrokes. We note that the sound of pressing a space key is distinct from that of other keys on a keyboard, so we can recognize them by audio signals. In this paper, we propose a novel and efficient method to find all keystrokes in the audio signals and distinguish the space keystrokes, which solves several problems in traditional ways. In addition, the number of words is large in a real situation, making it difficult to classify all of the words. Actually, the attacker does not need to know every word being typed by the target user, that is, the keywords in the dictionary are enough. Considering that the number of keywords could still be large, we design an ensemble model that divides the word set into subsets to decrease the number of classes for each submodel. This work is an extended version of our previous research[5] and mainly has the following contributions:

1) We proposed a method to recognize typed words on the basis of audio and accelerometer signals.
2) We proposed a model to recognize words on the word scale, different from the previously used scale of a single key or pair of keys.

3) We designed an ensemble model to decrease the number of classes for each submodel, considering that the number of words can be large.
4) We built a dataset containing 100 classes of words and 100 samples for each class of words, with the corresponding accelerometer and audio signals.

The main changes and extensions in this study include the following:

1) We replaced the keystroke recognition algorithm used in our previous work with a novel and efficient detection method that finds all keystrokes in the audio signals and distinguishes the space keystrokes.
2) We extended our dataset from one participant to four participants and ran more comparative experiments on the basis of this dataset.
3) We gave a simple discussion of how to prevent this type of attack by implementing our method.

The rest of this paper is organized as follows. In Sect. 2, we review the related works. In Sect. 3, we present the details of WatchLogger. In Sect. 4, we describe the dataset and settings of the experiments, as well as the experimental results. In Sect. 5, we discuss some issues in a real situation. Finally, we give a brief conclusion.

2. Related Work

Side-channel attacks are a class of physical attacks that try to dig up information from physical message leakages.[6] For mobile devices, the huge amount of sensitive information stored on them makes them more vulnerable to such attacks.[7] Many groups are attempting to perform recognition on the basis of the sounds produced during the input process. Shumailov *et al.*[8] proposed a method that can infer the text that users input into their smartphone by differentiating the waveform of tapping sounds. In the study by Zhu *et al.*,[9] acoustic signals were used for estimating the physical positions of keystrokes to infer the users' contents typed on a keyboard. In the study by Anand and Saxena,[10] a defense mechanism against keyboard acoustic attacks was proposed, wherein masking signals are added to leaking sounds to distort the original signals. We can see that if the sounds are contaminated by environmental noise, the accuracy of these classes of methods can decrease.

The accelerometer signals are another tool for side-channel attacks. De Souza Faria and Kim[11] proposed a method aimed at stealing passwords from an ATM keypad by analyzing the vibration generated when pressing the keys, using an accelerometer placed near the keypad. In the study by Wang *et al.*,[12] the passwords were inferred by the wrist-worn accelerometer that reproduces the trajectories of the user's hand. In the study by Maiti *et al.*,[13] the experimental results demonstrated that the wrist-worn motion sensors are very effective in narrowing the search space for unlocking combination locks. For the keystroke inference, Maiti *et al.*[1] proposed a framework to show how to infer the typed words using smartwatches, which utilizes accelerometer signals to recognize the directions of hand movements and infers the words from the series of directions. They also noted that the keystroke events can be detected by their sound. However, it was only conceptually mentioned in their paper so detailed implementation and results were lacking.

Different from the previous work, our method combines both sound and accelerometer signals to make the best use of the information from each signal. For the sound signal, since we only distinguish the space and nonspace keystrokes, the information contained in the signal of pressing a key is sufficient to classify these two types of keystrokes. For the accelerometer signal, we use the word-by-word basis because the movement of typing a word is more informative than pressing a single key. These make our system more robust with high performance.

3. System Design

The system design is as follows. The audio and accelerometer signals, which are simultaneously collected from a smartwatch, are input to the system. The signals are first preprocessed to remove the components that are not needed. Then, the keystroke detection module finds all the locations on the time axis where keystrokes occur on the basis of the audio signals. With these keystroke locations, the segmentation module can segment the accelerometer signals into frames that correspond to words. Finally, the trained classification model takes these frames as input and outputs the target words. Figure 1 shows the overview of WatchLogger.

3.1 Preprocessing

First, the simultaneously collected audio and three-axis accelerometer signals are preprocessed to remove the redundant parts from their waveform. Specifically for the audio signals, a three-order Butterworth high-pass filter is adopted to tackle the background noises made by surrounding equipment such as the fans in computers and air conditioners. By observing the spectrum of the audio signals, the frequency distribution of noises can be determined. Therefore, we can set the cut-off frequency of the filter as shown in Fig. 2(a). In this study, we set the cut-off frequency to 500 Hz. We also performed downsampling on audio signals to reduce the computational cost from the original 8000 to 320 Hz. For the accelerometer signals, we can see from Fig. 2(b) that the direct component caused by gravity and the low frequency trend caused by macro arm movements are mixed in. Since the information about typing activities is only reflected by finger and wrist micromovements, which are represented by the fluctuation in accelerometer waveform, the direct and low-frequency components are unwanted and adverse for recognition and should be removed. Wavelet decomposition is a useful method in this situation as it can decompose the signals into detailed and approximate parts that represent high- and low-frequency components, respectively, at several levels.[14] By removing the approximate parts and reconstructing signals using the detailed parts, the direct and low-

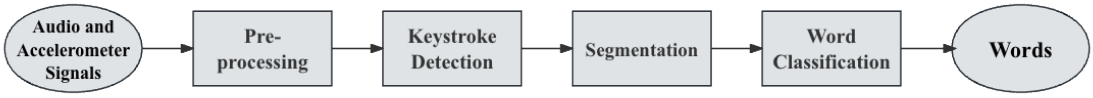
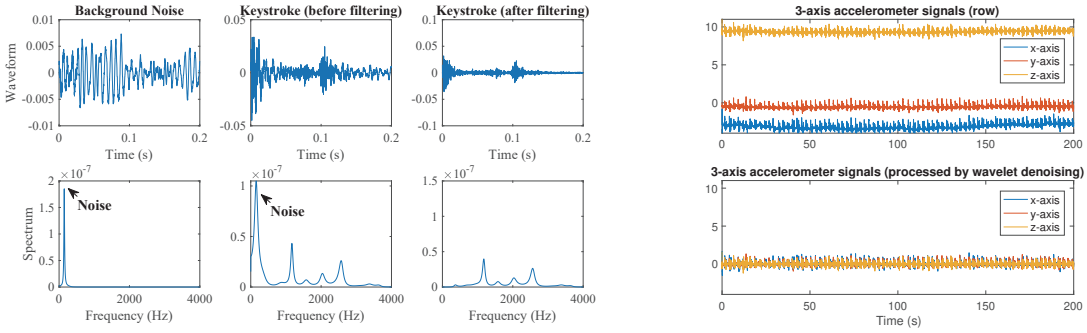


Fig. 1. Overview of WatchLogger.



(a) (b)

Fig. 2. (Color online) Preprocessing of (a) audio and (b) accelerometer signals.

frequency components can be suppressed, as shown in Fig. 2(b). In this study, we adopted the db4 wavelet[15] to decompose the signals into six levels and remove the approximate part at level 6.

3.2 Audio-based keystroke detection

In the keystroke detection module, the targets are defined as finding all keystrokes on audio signals and recognizing the space keystrokes among all keystrokes. Since the sound of pressing a space key is different from that of pressing other keys, it is possible to train a model to distinguish them from the audio signals. The traditional way, which was also adopted in our previous study,[5] is to apply a sliding window to group the audio signals into frames and use a trained three-class classifier on each frame to recognize the sounds of nonspace keystrokes, space keystrokes, and frames other than keystrokes. However, the previous work was focused on the accuracy of classification results of all frames while the positions and numbers of keystrokes were minimally considered. In fact, it is difficult for the sliding-window-based method to precisely locate keystrokes on audio signals even if the classifier has high accuracy. The main reason is that the intervals between two consecutive keystrokes can be short if the user is typing rapidly; in this situation, the sliding window always contains keystrokes and the classifier tends to predict a positive result. Therefore, the actual positions and numbers are difficult to evaluate. In addition, the sliding-window-based method also has other problems such as the multiclass and window size problems,[16] making it difficult to build a reliable module for our system.

In this paper, we propose a novel and efficient method to find all keystrokes in the audio signals and distinguish the space keystrokes. This method is inspired by the classic object detection model You Only Look Once (YOLO),[17] which predicts positions, sizes, and classes of objects in an image in one process. We modify YOLO for keystroke detection on audio segments in a simpler way than in the object detection version. As shown in Fig. 3, we divide the input audio segment into a $1 \times S$ grid. The grid cell that contains the center of a keystroke will be used for predicting that keystroke. Each grid cell will predict the values of p, c, and l, where p is the probability that this cell contains a keystroke and c is the conditional probability of each class

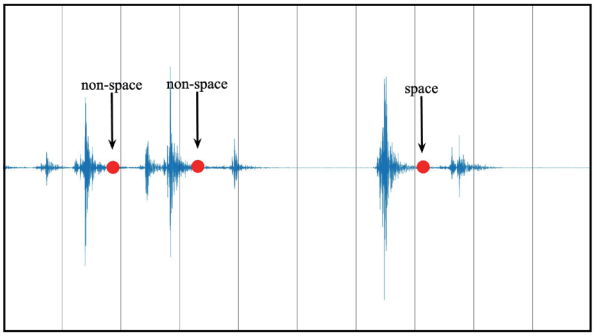


Fig. 3. (Color online) Grid segmentation of audio signals.

considering that this cell contains a keystroke. Here, it is a binary classification of space and nonspace keys. l ranges from 0 to 1, used to represent the relative location of the keystroke in a grid cell. Note that the length of keystrokes is not predicted because it does not provide any information for our system. The loss function L can be expressed as

$$L = \sum_{i=1}^{S} \xi_i^{obj} (l_i - \hat{l}_i)^2 + \alpha \sum_{i=1}^{S} \xi_i^{obj} (p_i - \hat{p}_i)^2 + \beta \sum_{i=1}^{S} \xi_i^{noobj} (p_i - p_i)^2 + \gamma \sum_{i=1}^{S} \xi_i^{obj} \sum_{j \in classes} (c_i(j) - c_i(j))^2, \quad (1)$$

where ξ_i^{obj} represents the condition that the i-th grid cell contains a keystroke, whereas ξ_i^{noobj} represents no keystroke. We apply three parameters to control the weight of each term: $\alpha = 5$, $\beta = 0.5$, and $\gamma = 5$.

The network architecture that we adopted in this study is a modified VGG model that was originally popular in 2D image classification.[18] It contains a stack of convolutional and max pool layers, and ends with fully connected layers. We changed all convolutional kernels to be 1D to fit the signal processing. For the output layer, the size of the tensor is $1 \times S \times 3$, where for S grids, each grid cell needs to predict three values. In this work, we set S to be 10.

3.3 Segmentation

After the keystroke detection, the positions of keystrokes and whether they are space keystrokes or not are known to the system. Under the assumption that we only consider space keystrokes as the delimiters in the target sentences, the accelerometer signals can be segmented into frames that represent the corresponding words by finding all space keystrokes and extracting frames between every two consecutive space keystrokes, as shown in Fig. 4. We also performed data augmentation to generate more frames for training. The method involves sliding the frame window forward and backward slightly on the accelerometer signals while determining the boundaries of a segmentation, as long as the word is not out of the boundaries. Here, we set the sliding size to 60 ms and two steps for both left and right, which augments the training set to be five times larger. Moreover, the number of keystrokes between two consecutive space

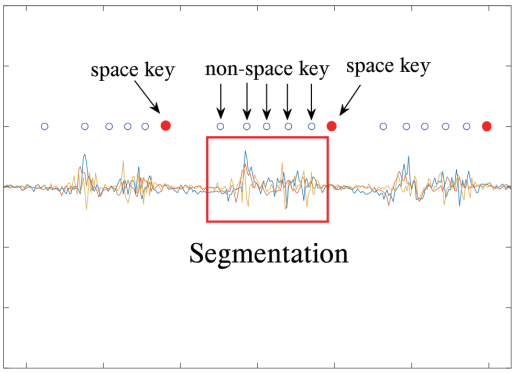


Fig. 4. (Color online) Example of how to segment accelerometer signals using space keystrokes.

keystrokes can be counted; in other words, the number of characters in a word or the word length len_w can be known after the segmentation stage.

3.4 Accelerometer-based word classification

The last stage is accelerometer-based word classification, where the inputs are the segmented three-axis accelerometer frames and the outputs are the corresponding words. The frames are first processed by feature extraction to obtain the expression of data as feature vectors, and then an ensemble model that consists of 10 similar submodels is used to predict the result words for each frame.

3.4.1 Feature extraction

We selected five commonly used features in this study, namely, mean, sum, median, standard deviation, and kurtosis, and we applied these five features to both the time and frequency domains across the accelerometer signals along all three axes. Moreover, we transformed the time and frequency frames into new frames using squaring and first-order difference, and applied the above five features to these generated frames to obtain more information from inputs. Therefore, there will be $5 \times 2 \times 3 \times 3 = 90$ features in total. In addition, len_w is also an apparent feature for word classification, considering that the value is known at this stage, as we discussed above. However, instead of putting len_w into the feature vector directly, we used it as the selector to choose a submodel in the ensemble model for each word, which we will discuss in the following subsection. All features will be processed by z-score standardization.

3.4.2 Ensemble classification

We can see from Fig. 5 that different words have slightly different patterns of hand movements. Therefore, it is possible to train a word classifier using the labeled accelerometer signals collected from the wrist-worn sensor. In this study, the classification model was chosen as a three-layer neural network with one hidden layer. Originally, we used the 90 features plus

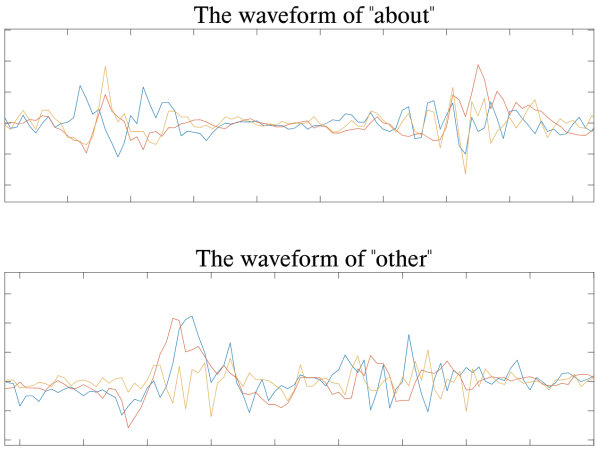


Fig. 5. (Color online) Waveforms of different words.

the additional feature len_w as the input and the words encoded by one-hot vectors as labels to train one model for word classification. However, there are a large number of different words in the vocabulary, and the number may still be large even if we only pick a subset of words for the target user; this can cause a decrease in classification accuracy compared with the result using a small dataset.[19] To address this potential issue, we proposed the ensemble classification model in which the original single model is split into n similar submodels, as shown in Fig. 6.

Intuitively, if the number of classes, n_c, is very large in a classification problem, it will be a good solution to split the large class set into groups to have a smaller n_c in each group and then perform classification within these groups as a tree structure, where the rule of how to split the class set is highly empirical. Fortunately, for the word classification task, the length of a word is a very efficient and off-the-shelf rule with which to address the problem. Specifically, the feature of len_w is removed from the input vector; then, the original model is split into n submodels $\{m_1, m_2, ..., m_n\}$, where $m_{i,i\in[1,n]}$ is trained by the samples $\{s_{i,1}, s_{i,2}, ..., s_{i,k}\}$, where all $len_w(s_{i,j})$, $i \in [1,n]$ and $j \in [1,k]$ are equal. In the training phase, we defined the model length $len_m(i)$ for the i-th model and picked all samples that meet $len_w(s_j) = len_m(i)$ from the dataset as training data. In the testing phase, the selector will match each sample s_i to the model m_j where $len_w(s_i) = len_m(j)$ as the input, and the ensemble model will adopt the output from m_j as the final output. By this method, n_c of each submodel can be reduced by n times compared with the original model if we assume that all submodels have the same n_c, and thus, the problem of having a large n_c can be alleviated to a certain extent. In this study, we set n to 10 and len_m from 3 to 12.

4. Experimental Results

In this section, we first introduce the dataset built for evaluating WatchLogger. Then, we list the parameters and settings of the experiment. Finally, we give the experimental results and analysis.

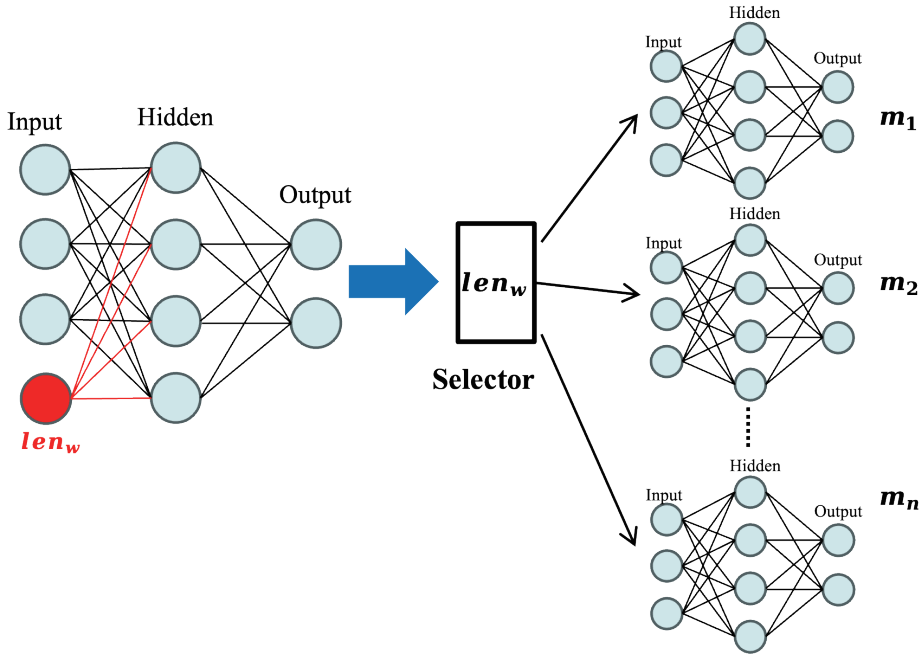


Fig. 6. (Color online) Process of how original model splits into ensemble model.

4.1 Data collection

To validate our method, we built WTW-100, the Wearable Typed Words dataset with 100 classes of words and 100 samples for each class of each participant, using data collected from four participants. We used the alps LEMT with Android 7.1.1 OS as the smartwatch that has a three-axis accelerometer with a sampling rate of 50 Hz and a microphone with a sampling rate of 8000 Hz. The application running on the smartwatch can simultaneously collect audio and accelerometer signals. The keyboard that we used for input is FILCO FILCKTLBT2-33, connected to a MacBook via Bluetooth. The reason why we did not use a MacBook keyboard is that the sound of the MacBook keyboard is relatively low and will be easily affected by noise. Considering that our target is to prove the feasibility of the method, we applied FILCO FILCKTLBT2-33, which produces a clearer sound, to build a dataset. To obtain the truth label, we ran a keylogger application on the MacBook to log all keystroke events that contain pairs of the key content and timestamp. The dataset was uploaded to GitHub and can be searched by name.

The data are collected as follows. First, we launch the application on the smartwatch and the keylogger on the MacBook. Then, the participant wears the smartwatch tightly. After that, the participant clicks the start button on the smartwatch and starts typing. When finishing typing all the required words, the participant clicks the stop button and the record is saved. Figure 7 shows the typing process. To simplify the process, for each record, the participant is asked to type the same words a certain number of times. Each participant is asked to type all 100 words 100 times, while participant 4 (P4) types each word 100 times in one record, and P1, P2, and P3 type each

Fig. 7. (Color online) Typing process.

word 100 times in three to four records. None of the four participants are native English speakers (one Chinese person and three Japanese persons). Specifically, for P4, the typing speed and rhythm of the same word are controlled to be as consistent as possible to simulate the ideal typing situation of native English speakers, while for other participants, there is no limit on the typing speed or rhythm; therefore, it is less consistent for the same word collected from P1 to P3.

Additionally, the words of WTW-100 have lengths from 3 to 12 letters and 10 words for each length. Considering that short words such as “a” and “of” are less informative and less distinguishable, we removed words with lengths of one and two letters. We selected the most commonly used words of each length as the dictionary of our dataset, on the basis of the word frequency derived from the Google Web Trillion Word Corpus.

4.2 Experimental settings

To evaluate the performance of the keystroke detection module, we analyzed three results, namely, the results of keystroke detection, classification, and position. For detection results, we used accuracy, precision, recall, and F1 score to evaluate the performance of finding keystrokes, eliminating the bias caused by the imbalance between the numbers of grid cells with and without a keystroke. For classification results, we evaluated the same values as those of detection because the numbers of space and nonspace keystrokes are also unbalanced. For positions, we evaluated the average absolute error (*AAE*) in the time axis.

For the structure of the VGG model in the keystroke detection module, the input is downsampled audio segmentation with a length of 1 s. The structure of convolutional layers is [64, ’M’, 128, ’M’, 256, 256, ’M’, 512, 512, ’M’], where the numbers refer to the number of out channels and ’M’ means the application of max pool. The kernel size is set to 7, and each layer adopts the ReLU activation function. The numbers of nodes of fully connected layers are 512, 1024, and 30, and each layer has a dropout rate of 0.5 and is trained for 100 epochs with a learning rate of 0.0001.

For the word classification module, we evaluated the accuracy of classification results. We also compared the performance of the original single model with that of the ensemble model to show the effectiveness of our ensemble method. For the single model, we used a three-layer neural network in which the numbers of nodes are 31, 256, and 100. For the ensemble model, we

used 10 independent three-layer neural networks that have layer sizes of 30, 512, and 10 nodes. These models were all trained for 2000 epochs with a learning rate of 0.001. All the models have a ReLU activation function in the hidden layer. The cross-entropy loss is adopted as the loss function.

We used PyTorch 2.0.1 to implement all models and train them on a Windows 11 PC with NVIDIA GeForce RTX 4070 Ti GPU, and we applied individual and generalization tests to evaluate the training results. For the individual test, each model was evaluated on the dataset collected from one participant through *k*-fold cross-validation that divides the samples into k parts and then uses $k-1$ parts for training and one part for testing. This process was rotated k times for each participant. In this paper, we set k to 10 and used the average as the result. For the generalization test, to evaluate the generalization performance of our method, we applied leave-one-out cross-validation where the data of one participant are separated as the test set and those of the others are the training set.

4.3 Experimental results

4.3.1 Individual test

Table 1 shows the results of keystroke detection in the individual test. The average accuracy, precision, recall, and F1 score of detection are 98.31, 95.98, 98.98, and 0.9745%, respectively, indicating a high performance of our model in accurately detecting keystrokes on audio signals. For classification, even though the numbers of space and nonspace keystrokes are highly unbalanced (about 1:7.5), the results still reached 99.62, 98.77, 98.32, and 0.9855%, respectively. In addition, the average *AAE* of position is 10.41 ms, which is a considerable result for detecting the locations of keystrokes. Figure 8 illustrates one of the keystroke detection results for an audio segment.

Table 2 shows the result of word classification for each length using ensemble models in the individual test. The average accuracy of the ensemble models of all participants reached 79.76%. Figure 9 shows the results of the single model and ensemble model tests. We can see that, compared with the single model, the ensemble model has an increase in accuracy of 8.51%, showing a satisfactory performance in word classification. Another interesting phenomenon is

Table 1
Results of keystroke detection in individual test.

		P1	P2	P3	P4	Avg
Detection	Accuracy (%)	98.12	98.34	98.03	98.75	98.31
	Precision (%)	94.92	95.37	96.59	97.05	95.98
	Recall (%)	98.93	99.18	99.13	98.66	98.98
	F1 score	0.9688	0.9724	0.9784	0.9785	0.9745
Classification	Accuracy (%)	99.39	99.69	99.74	99.67	99.62
	Precision (%)	98.42	99.16	98.95	98.56	98.77
	Recall (%)	97.70	98.41	98.38	98.80	98.32
	F1 score	0.9806	0.9878	0.9866	0.9868	0.9855
Position	*AAE* (ms)	9.95	10.51	10.37	10.82	10.41

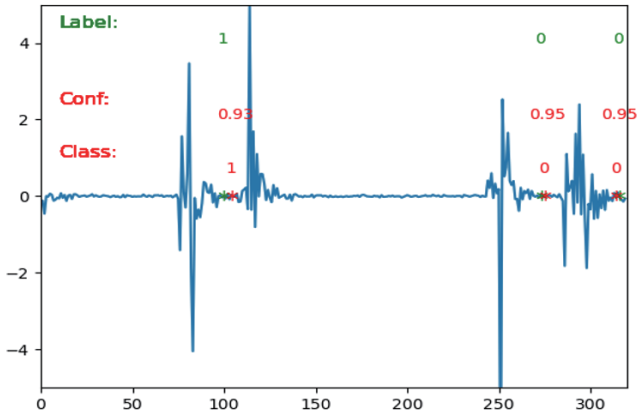


Fig. 8. (Color online) Example of keystroke detection results.

Table 2
Accuracy (%) of word classification of each length using ensemble models in individual test.

	Len 3	Len 4	Len 5	Len 6	Len 7	Len 8	Len 9	Len 10	Len 11	Len 12	Avg
P1	67.74	73.19	69.86	73.96	72.83	72.32	72.50	78.11	78.34	73.50	73.24
P2	64.91	70.06	71.06	78.17	76.76	77.45	81.15	79.12	79.65	84.20	76.25
P3	64.31	75.86	70.46	74.07	76.91	73.09	75.51	75.55	76.39	85.19	74.73
P4	93.36	94.83	91.39	93.98	96.72	95.52	94.47	95.74	96.01	96.17	94.82
Avg	72.58	78.49	75.69	80.05	80.81	79.60	80.91	82.13	82.60	84.77	79.76

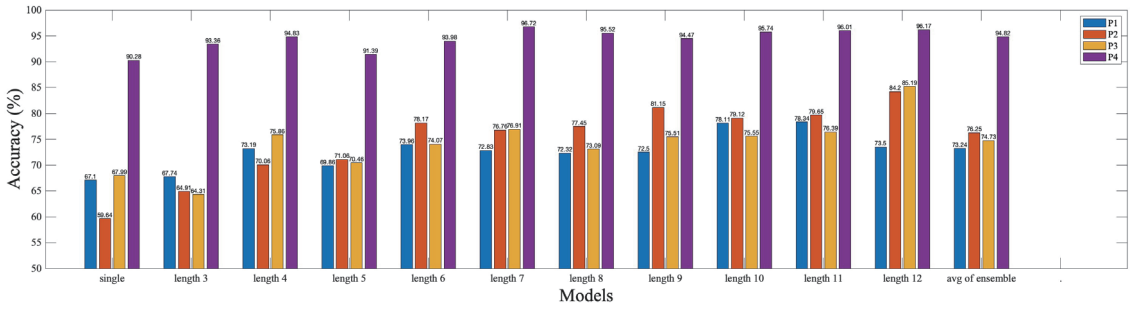


Fig. 9. (Color online) Results of single model and ensemble model tests.

that the accuracy of the ensemble model tends to be higher for longer words than for shorter words. The main reason for this could be that the longer words have more combinations of hand movements and are thus more distinguishable than the shorter words.

Figure 10 shows the results of top five words that have the highest and lowest error rates. We can see once again that the words with the highest error rate tend to have a smaller length than those with the lowest error rate owing to the different amounts of information contained in them. Specifically, the word “can” has the highest error rate, and it is misrecognized as “new” most of the time because there is only one pair of keys on the left side and the pattern of left-hand movement is both right to left for “ca” and “ew”, which contains minor information for classification. Such an error is inevitable in the acceleration-based system because the similarity of words commonly exists, but the results still show a certain ability to classify these words.

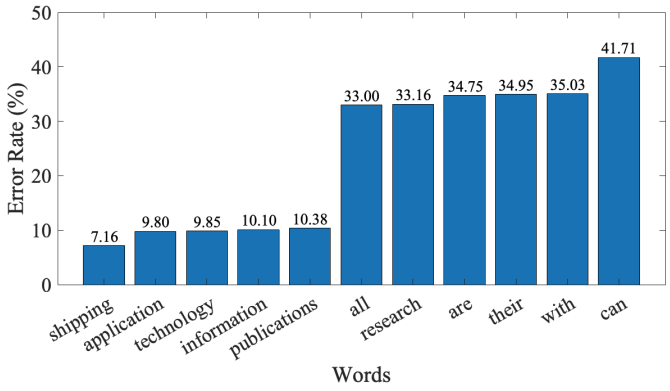


Fig. 10. (Color online) Error rates of top five words that have the highest and lowest error rates.

In addition, it can be observed that the result of P4 is significantly better than those of the other participants. As mentioned above, for P4, the typing speed and rhythm of the same word were controlled to be as consistent as possible, while the others were not. Intuitively, an inconsistent typing habit would make the result of word classification worse owing to the different distributions of some of the training and test sets. To explain this phenomenon, we verified our dataset by calculating the time variance of the same words being typed by each participant. The average time variances of P1, P2, P3, and P4 were 0.1548, 0.2315, 0.0641, and 0.0230 s^2, respectively, where P4 had a much smaller time variance than the others. Another possible reason why the performance was better for P4 than for the others is that all 100 samples of a word were collected from one typing experiment in the former case, while in the latter case, samples were collected from multiple experiments on different days; this could introduce more different distributions of data because the conditions (e.g., the sitting posture) for each experiment could be different. Nevertheless, the average accuracy for P1 to P3 still reached 74.74%, indicating that our method has satisfactory performance in some real-life situations.

4.3.2 Generalization test

Tables 3 and 4 show the results of keystroke detection and word classification in the generalization test, respectively. Here, the result for P1 represents the result where the data from P1 were used as the test set and those from the others were used as the training set. Compared with the results of the individual test, the results of keystroke detection and word classification in the generalization test decreased generally, where keystroke detection had a smaller decrease in performance because the sound of keystrokes only had subtle changes among different participants, whereas word classification had a significant decrease in performance, indicating that our method is not suitable for the situation where a model is pretrained and applied to a new user. This is reasonable because the motion pattern of typing a word would be unique for each person. The best application of our method in the real world is collecting data from the target user, which will be discussed in the next section.

Table 3
Results of keystroke detection in generalization test.

		P1	P2	P3	P4	Avg
Detection	Accuracy (%)	87.99	89.80	85.19	81.00	86.00
	Precision (%)	79.63	80.97	81.92	64.68	76.80
	Recall (%)	91.14	91.04	88.18	86.98	89.34
	F1 score	0.8500	0.8571	0.8493	0.7419	0.8260
Classification	Accuracy (%)	95.51	95.49	91.98	86.75	92.43
	Precision (%)	79.36	80.86	87.43	44.99	73.16
	Recall (%)	84.94	83.06	40.55	62.44	67.75
	F1 score	0.8206	0.8195	0.5540	0.5230	0.7035
Position	*AAE* (ms)	15.27	15.15	16.57	16.80	15.95

Table 4
Accuracy (%) of word classification of each length using ensemble models in generalization test.

	Len 3	Len 4	Len 5	Len 6	Len 7	Len 8	Len 9	Len 10	Len 11	Len 12	Avg
P1	31.15	21.94	26.94	27.72	28.58	28.60	27.46	27.17	27.34	26.63	27.35
P2	27.00	27.56	22.31	30.86	31.35	32.84	38.63	30.81	35.09	33.95	31.04
P3	28.80	29.76	30.97	26.81	35.38	31.74	35.39	30.86	27.69	30.54	30.79
P4	27.43	32.46	40.50	27.16	31.92	27.52	38.26	47.48	36.04	31.78	34.06
Avg	28.60	27.93	30.18	28.14	31.81	30.17	34.94	34.08	31.54	30.73	30.81

5. Discussion

In this section, we discuss some issues that are inevitable to WatchLogger, and we explain how our method could be made more reasonable when it is implemented in a real situation.

5.1 Single-hand problem

As we know, typing involves both hands, but the smartwatch can only record one hand, causing missing information for typed word recognition. If different words have the same motion patterns of the watch-worn hand, it will be difficult to recognize these words. For example, “can” and “cap” have the same pattern of “ca” for the left hand. To avoid the single-hand problem, on the one hand, the dictionary can be carefully selected by the attacker to avoid such collisions. On the other hand, people tend to have a distinct typing preference (e.g., typing rhythm) for each word, so the word could be distinguishable even if the two words have the same pattern. However, in some cases, the collision will be inevitable, and the system may need some more information resources. Sound might be a possible information resource, but the difference in the sound of each key is too subtle to distinguish, and we need to investigate how to find the best representation of sound signals so that the information of typed words can be dug out. We also need to study how to fuse sound signals and acceleration. We plan to acquire more information about typing activity from more potentially available sensors in our future research.

5.2 Data acquisition problem

In a real situation, how to acquire data for training is another problem for WatchLogger. Intuitively, we can build the dataset using the data obtained from several participants and attack all target users with it. However, different people have different preferences for typing motions, and the information the attacker wants to steal from each user may also be different. Therefore, a personalized dataset will be more efficient. Here is a potential method. First, the attacker disguises WatchLogger as a normal application and lures the target user to install it. Then, the attacker can trick the user into typing some given words. For example, it can be disguised as an online part-time job that asks the employees to type up a handwritten manuscript. The attacker can select a set of keywords as a dictionary for the task. In this way, the attacker can build a dataset for this target user.

5.3 Attack prevention

After discussing the full details of our attack method, it is critical to provide corresponding solutions to prevent such attacks. There are two types of solution: system side and user side. For the system side, assuming the malicious application cannot be recognized, the smartwatch system should block all access permissions to sensor data temporarily when typing activities are detected. The keystroke detection of our method is a good way of detecting typing activity. The system can run the detection program periodically (e.g., every hour) to sample one segment of audio and detect keystrokes on it, ensuring both accuracy and cost. For the user side, users should be aware that their typing data is also sensitive information just like fingerprints. As shown by the results of the generalization test, the motion pattern of typing a word could be unique for each person; therefore, it is difficult to achieve such attacks if the attacker cannot acquire any typing data from the target user. We should all ensure that our typing data will not be leaked easily, as discussed in Sect. 5.2.

6. Conclusions

In this paper, we proposed WatchLogger, a method that uses both audio and accelerometer signals on a smartwatch to recognize English words being typed by a target user. WatchLogger applies a novel and efficient audio-based keystroke detection method to find and distinguish all space and nonspace keystrokes from their sounds; this was inspired by the YOLO model in 2D image object detection, and we modified it to fit our 1D audio signal detection task. Knowing the positions of all keystrokes, WatchLogger uses space keystrokes to segment accelerometer signals into frames that represent target words. In addition, to classify words from accelerometer signals and deal with the problem caused by too many words, we proposed an ensemble classification model that splits the model and word set into submodels and subsets, where each submodel has fewer classes for classification.

To evaluate WatchLogger, we built the WTW-100 dataset that contains 100 classes of words and 100 samples for each class, using the data collected from four participants. The experimental

results showed accuracies of 98.31 and 99.62%, and F1 scores of 0.9745 and 0.9855 for keystroke detection and classification, respectively, and an accuracy of 79.76% for word classification, indicating the satisfactory performance of WatchLogger. Finally, we discussed some issues of implementing WatchLogger, including the single-hand and data acquisition problems, and explained the feasibility of WatchLogger in a real situation. We also gave a brief discussion about preventing attacks on the basis of our method, for both the system and user sides.

Acknowledgments

This work was partially supported by JST SPRING (Grant Number JPMJSP2136) and JSPS KAKENHI (Grant Number JP19KT0020).

References

1 A. Maiti, O. Armbruster, M. Jadliwala, and J. He: Proc. 11th ACM Asia Conf. Computer and Communications Security (ACM, 2016) 795–806. https://dl.acm.org/doi/pdf/10.1145/2897845.2897905
2 Y. Berger, A. Wool, and A. Yeredor: Proc. 13th ACM Conf. Computer and Communications Security (ACM, 2006) 245–254. https://dl.acm.org/doi/pdf/10.1145/1180405.1180436
3 P. Marquardt, A. Verma, H. Carter, and P Traynor: Proc. 18th ACM Conf. Computer and Communications security (ACM, 2011) 551–562. https://dl.acm.org/doi/pdf/10.1145/2046707.2046771
4 J. Harrison, E. Toreini, and M. Mehrnezhad: Proc. 2023 IEEE European Symp. Security and Privacy Workshops (IEEE, 2023) 270–280. https://arxiv.org/pdf/2308.01074
5 G. Li, Y. Arakawa, Y. Nakamura, H. Choi, S. Fukushima, and W. Wang: Proc. 2023 14th Int. Conf. Mobile Computing and Ubiquitous Network (ICMU, 2023) 1–6. https://doi.org/10.23919/ICMU58504.2023.10412218
6 F. X. Standaert: Secure Integrated Circuits and Systems (Springer, Boston, 2010) pp. 27–42. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=01454c39dc7090e216c9a43562b493728d0a93fe
7 R. Spreitzer, V. Moonsamy, T. Korak, and S. Mangard: IEEE Commun. Surv. Tutorials **20** (2017) 465. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8141882
8 I. Shumailov, L. Simon, J. Yan, and R Anderson: arXiv:1903.11137 (2019). https://arxiv.org/pdf/1903.11137
9 T. Zhu, Q. Ma, S. Zhang, and Y. Liu: Proc. 2014 ACM SIGSAC Conf. Computer and Communications Security (ACM, 2014) 453–464. https://dl.acm.org/doi/pdf/10.1145/2660267.2660296
10 S. A. Anand and N. Saxena: Proc. Financial Cryptography and Data Security: 20th Int. Conf. (FC, 2016) 346–364. https://fc16.ifca.ai/preproceedings/21_Anand.pdf
11 G. de Souza Faria, and H. Y. Kim: IEEE Trans. Inf. Forensics Secur. **8** (2013) 1221. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6525326
12 C. Wang, X. Guo, Y. Chen, Y. Wang, and B. Liu: IEEE Trans. Mob. Comput. **17** (2017) 646. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8006273
13 A. Maiti, R. Heard, M. Sabra, and M. Jadliwala: Proc. 11th ACM Conf. Security and Privacy in Wireless and Mobile Networks (ACM, 2018) 111–122. https://dl.acm.org/doi/pdf/10.1145/3212480.3212498
14 C. U. I. Huimin, Z. Ruimei, and H. O. U. Yanli: Physics Procedia **33** (2012) 1354. https://doi.org/10.1016/j.phpro.2012.05.222
15 C. Torrence and G. P. Compo: Bull. Am. Meteorol. Soc. **79** (1998) 61. https://doi.org/10.1175/1520-0477(1998)079%3C0061:APGTWA%3E2.0.CO;2
16 R. Yao, G. Lin, Q. Shi, and D. C. Ranasinghe: Pattern Recognit. **78** (2018) 252. https://arxiv.org/pdf/1702.06212
17 J. Redmon, S. Divvala, R. Girshick, and A. Farhadi: Proc. IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2016) 779–788. https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf
18 Q. Guan, Y. Wang, B. Ping, D. Li, J. Du, Y. Qin, and J. Xiang: J. Cancer **10** (2019) 4876. https://doi.org/10.7150%2Fjca.28769
19 J. Deng, A. C. Berg, K. Li, and L. Fei-Fei: Proc. Computer Vision–ECCV 2010: 11th European Conf. Computer Vision (ECCV, 2010) 7184. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=9800e3c3394c569be83379ee2ebe3424e09c2919