ECHOES: input sensing and rEconstruCtion model utilizing CNN and LSTM witH mObilE Sensor data

Hongyi Hao
Shanxi Institute of Technology
Yangquan, China
Email: lanyi_adict@outlook.com

Abstract* — *The widespread integration of sensors such as accelerometers and gyroscopes in smartphones has significantly enhanced user experience, but also introduced new vulnerabilities exploitable by side-channel attacks. These attacks can infer user input by analyzing motion patterns generated during touch interactions. To mitigate this threat and improve input recognition accuracy, we propose ECHOES, a side-channel analysis framework designed to recover user input on touchscreen keyboards. ECHOES comprises two key modules: (i) a sensing module that collects inertial sensor data and virtual memory change patterns, employing convolutional neural networks to detect the user's motion state and classify touch interactions into predefined keyboard regions; and (ii) a language inference module that leverages the click region sequence and natural language processing to reconstruct the user's intended input. To further improve localization under spatial ambiguity, the keyboard is partitioned into five regions to constrain candidate outputs. Experimental results demonstrate that ECHOES achieves 94.01% accuracy in motion state recognition and 91.63% accuracy in input restoration, underscoring its potential for exposing security risks in mobile input systems.

Keywords—Smartphone sensors, Activity recognition, Keystroke inference, Deep learning.

I. INTRODUCTION

With the rapid advancement of science and technology, modern smartphones are increasingly equipped with a wide range of sensors, including GPS [2], grip sensors, linear vibration motors [3], light sensors, accelerometers [4], gyroscopes, and pressure sensors [1]. These embedded components enhance interactivity and enable functionalities such as navigation, health monitoring, automatic screen control during calls, and activity tracking [5]. While these sensors significantly improve user experience, they also introduce novel vulnerabilities that can be exploited by side-channel attacks.

Traditional side-channel attacks typically rely on acoustic signals or electromagnetic radiation to infer keystrokes. However, with the growing volume of sensitive information entered via smartphones, attackers have begun leveraging onboard motion sensors. For instance, applications such as TouchLogger [6] utilize accelerometers and gyroscopes to capture minute vibrations from the screen, analyzing parameters such as finger movement direction and pressure to infer user input.

During text entry, users frequently make mistakes due to distractions, hand instability, or inaccurate visual targeting. If the system can autonomously recover the intended input without requiring user correction, it not only enhances typing accuracy but also mitigates the effectiveness of side-channel attacks—incorrect input data becomes unusable to potential adversaries. Consequently, improving the recognition of keyboard input under various contextual conditions has become an urgent research direction.

Recent studies have explored input recognition from diverse perspectives, including acoustic signals, deep neural networks, and localization techniques. For example, GMM-UBM and DBN models have been employed to enhance keystroke accuracy [7], while audio classification methods improve keystroke dynamics using models such as Extreme Gradient Boosting and Multi-Layer Perceptrons [8]. Additionally, Keystroke Dynamics [9] combines CNN and RNN architectures to achieve robust biometric recognition of typing behavior.

In this paper, we propose a novel side-channel analysis method that leverages shared memory access patterns [10]. By correlating changes in an application's shared memory with top-level window events in the system stack, we infer user motion states—an essential precursor to accurately locating touch positions. Recurrent neural networks are then used to identify click locations, enhanced by a dynamic temporal regularization algorithm [11] for fine-grained time series matching. The detection accuracy is further improved by analyzing longer input sequences, which generate more touch events and thus offer richer temporal features.

To address the spatial ambiguity inherent in dense virtual keyboards, we divide the keyboard into five distinct regions, reducing the complexity of precise click localization. Finally, natural language processing techniques [12] are applied to recover word sequences and reconstruct full sentences. Together, these contributions aim to improve the accuracy of smartphone text input while exposing critical security implications of sensor-based side-channel vulnerabilities.

II. SYSTEM DESIGN OF ECHOES

A. Data acquisition

Data collection is carried out by applications installed on Android smartphones. We selected 10 volunteers to conduct experiments on five mobile phones, including the Vivo X9. The

motion state includes static state and motion state. Static state includes: standing, sitting, and lying. Movement includes walking, upstairs, and downstairs [13].

The capture screen is vertical when entering the contents of letters. The volunteer held the smartphone in his left hand and tapped with the index finger of his right hand. According to the experiment, the average rate of entry for adults aged 27 was 41.01 words per minute [14], so the volunteers entered at a rate of two letters per second. During walking, the walking speed was one step per second. The data collection time for each phone is three minutes. After data collection is completed, the data will be saved as a comma delimited file.

B. Data pre-processing

After the data are completed, the data are preprocessed first. To make the extracted signal smooth enough and provide convenience for subsequent work, the moving average filter [15] is used for smoothing. Since the clock speed is faster than the dynamic behavior frequency under motion, the click event period is calculated to be 1.5 seconds. Multiple click events can be calculated in one click event cycle, the collected samples are aggregated, and several features of each click are extracted,the purpose is to obtain as many features as possible about clicks. x', y', z', m' represents the first derivative of triaxial acceleration and Euclidean norm[16] respectively. The gyroscope is the same calculation method. The first derivative

is the change rate at a certain time, corresponding to a certain period of a click. Finally, the eigenvalues are summarized as Tab. 1.

C. Model training

1) Motion sensing:

Sensor data is time series data that continuously varying, and convolutional and pooling layers can effectively handle such varying data, This study uses a convolutional neural network to perform convolutional operations [18] instead of matrix multiplication (as shown in Fig. 1).

In the Equ.1 , if $x_i^0 = [x_1 \dots, x_n]$ is the input vector of accelerometer and gyroscope sensor data, N is the number of values of each window, and the output of the first convolution layer is :

$$c_j^1 = \sigma\left(b_j^1 + \sum_{m=1}^{M} w_m^{1,j}\, x_{l+m-1}^{0,j}\right) \quad (1)$$

Similarly, the output of the l convolution layer can be calculated as follows:

$$c_i^{l,j} = \sigma\left(b_j^l + \sum_{m=1}^{M} w_m^{l,j}\, x_{l+m-1}^{l-1,j}\right) \quad (2)$$

$$p_i^{l,j} = \max_{r \in r}\left(c_{i^*t+R}^{l,j}\right) \quad (3)$$

Among them, c is the activity category, L is the index of the last layer, and N_C is the total number of activity categories.

Forward propagation is performed using the equation (1) - (4), which provides us with network error values. The training weight updating and error cost minimization are accomplished by the stochastic gradient descent (SGD) of the training data of small-batch sensors. The backward propagation through the full connection layer is calculated by :

$$\frac{\partial E}{\partial w_{ij}^l} = y_i^l \frac{\partial E}{\partial x_j^{l+1}} \quad (4)$$

Where E is the error/cost function, $y_i^l = \sigma\left(x_i^l + b_{ij}^l\right) w_{ij}^l$, w_{ij}^l is the weight of the unit u_i^l from layer l to layer $l + 1$, x_j^{l+2} is the total input of the unit u_j^{l+1}. The backpropagation of adjusting the weight of the convolution layer is accomplished by calculating the gradient of the weight:

$$\frac{\partial E}{\partial w_{ab}} = \sum_{i=0}^{N-M-1} \frac{\partial E}{\partial x_{ij}^l} y_{(i+a)}^{l-1} \quad (5)$$

where $y_{(i+a)}^{l-1}$ is a nonlinear mapping function, which is equal to $\sigma\left(x_{(i+a)}^{l-1}\right)$, and the increment $\frac{\partial E}{\partial x_{ij}^l}$ is equal to $\frac{\text{partialE}}{\partial y_{ij}^l} \sigma'\left(x_{ij}^l\right)$.

	Mean value
std	Standard deviation
mad	Median absolute deviation
max	Largest value in array
min	Smallest value in array
sma	Signal magnitude area
energy	Energy measure.Sum of the squares divided by the number of values.
iqr	Interquartile range
entropy	Signal entropy
arCoeff	Autoregression coefficients with Burg order equal to 4
correlation	correlation coefficient between two signals
maxInds	index of the frequency component with largest magnitude
skewness	skewness of the frequency domain signal
kurtosis	kurtosis of the frequency domain signal
bandsEnergy	Energy of a frequency interval within the 64 bins of the FFT of each window.
angle	Angle between to vectors.

Table 1: The proportion of attack types in the test set and the training set.

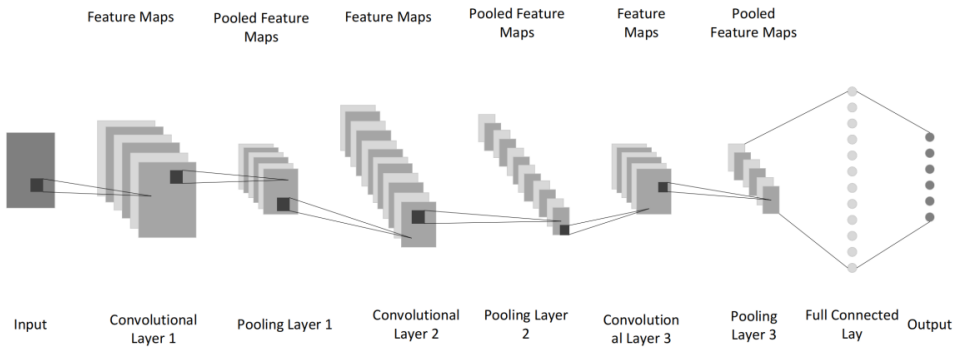

Figure 1: HAR(Human Activity Recognition) Convolutional neural networks.

Repeat forward and backward propagation until they stop standard is satisfied.

a) Regularization

Very large weights make it easy for the weight vector to fall into local minimums because gradient descent only changes the direction of optimization slightly. Weight attenuation is a regularization method, for each weight, the penalty term $\lambda \sum_w W^2$ is added to the cost function:

$$E = E_0 + \lambda \sum_w w^2 \quad (6)$$

A new cost function is generated, and the learning rule becomes:

$$w_i = (1 - \eta\lambda) w_i - \eta \frac{\partial E_0}{\partial w_i} \quad (7)$$

Where $1 - \eta\lambda$ is the weight attenuation factor.

Let $v = [v_1, v_1, \ldots, v_k]$ be a speed variable, one for each weight variable. The gradient descent update rule becomes:

$$v \to v' = \mu v - \eta \nabla E \quad (8)$$

$$w \to w' = w + v' \quad (9)$$

Where μ is the momentum coefficient.

Dropout [19] is equivalent to training many different neural networks. Networks with different architectures will over-fit in different ways, but their average results will effectively reduce over-fitting [20].

Before adjusting the pool size, this article only uses a simple softmax classifier. On the other hand, we switch to a multi-layer perceptron during subsequent runs so that performance improvements can be displayed and final changes merged into the architecture.

b) Hyperparameter

To evaluate the influence of the hyperparameters on the network performance when the motion state changes, the

1) Click event detection:

When detecting click events, the keyboard interface cannot be called directly to obtain click information. Most keyboards on the Android platform will appear with a floating animation effect on the top of the stack after the keystroke is completed. This animation is realized by Popwindows. The display and disappearance of floating window animation will affect the change of virtual memory size. Therefore, the occurrence of click events can be judged by detecting the size change of virtual memory.

As shown in the Fig. 2, it can be extended to the left and right at the midpoint of the peak of the virtual memory at an equal time interval. Time interval is the user's typing speed.

2) Click region detection:

After the click event is detected, the location of the click can be judged by the sensor data. As shown in the Fig.3, after clicking on the upper left corner and lower right corner of the mobile phone, the mobile phone screen will deviate in the direction of the force. The greater the click, the greater the offset angle.

To improve the recognition accuracy, the keyboard is divided into the following areas (as shown in Fig. 5).

3) Word restoration:

After determining the click position, the click position can be determined as a sequence. After the following algorithm, the input words can be inferred (as Algorithm 1).

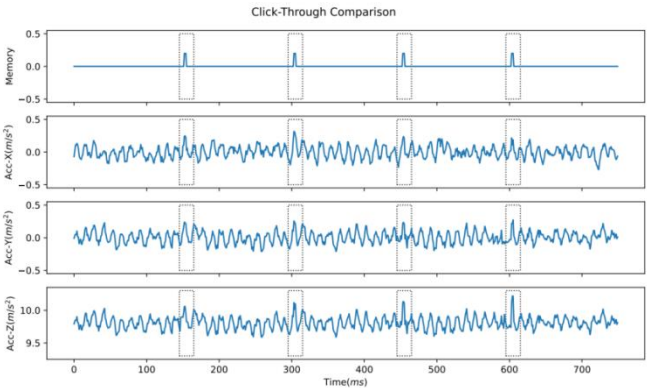

Figure 2: Memory changes after clicking on the keyboard.

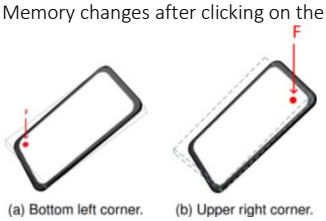

Figure 3: Tilt caused by clicking on different positions.

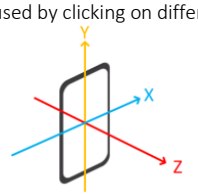

Figure 4: Virtual coordinate axis.

Figure 5: Keyboard area distribution.

Algorithm 1 Number Sequences to Letter Conversion

```
Input: Digits:String
Output:Letter combinations
table = ["abc", "def", "ghi", "jkl", "mno"]
list = []
q = []
q.append(" ")
while q is not empty do
    s = q.pop(0)
    if len(s) == Length then
            list.append(s)
    else
            val = table[int(Digits[len(s)])]
            for each character in val do
                q.append(s + character)
            end for
        end if
    end while
        return list
```

The algorithm can output a combination of letters in the keyboard partition after inputting a string composed of numbers [21].

4)Sentence restoration:

Text generation in natural language processing is applied to recover the input content. For the text generation task, the process is shown in Fig. 6). In this particular study, the focus is on training a word-level language model. The dataset is selected without restriction or misdirection, as this is a self-supervised learning method [22].

Following the selection of the dataset, all punctuation marks were removed during the process of data preprocessing, and the word length was limited to seven letters. The total number of training words was also limited to 100,000 words in order to save computational space and reduce time. In each option configuration, text generation employs a many-to-one structure.A context vector is only the input of the training model.

In this paper, a bidirectional LSTM with 256 hidden units is employed, utilizing long-term and short-term memory neural networks [23] .Instead of considering the seed word, the previously generated word is considered. The principal methodologies employed for the extraction of context vectors encompass importance sampling and word clustering[24].

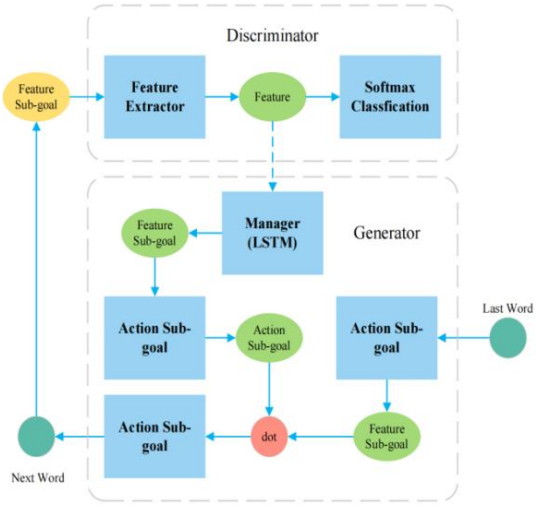

Figure 6: Sentence Restoration Method.

III. EXPERIMENT

A. *Activity Status Evaluation*

DTo better compare the performance of the state-aware model, the support vector machine [25], decision tree [26] , random forest , and K-nearest neighbor algorithm are used to compare with the convolutional neural network designed in this paper in many aspects. The results are as follows:

Activity status evaluation result include accuracy, recall, F value, and receiver operating characteristic curve (shown in Fig. 7). When recall rate is measured, the formula is recall $= \frac{TP}{TP+FN} = \frac{TP}{P}$. The larger AUC area indicates better performance. The following figure is the participant working curve of the motion perception model. After several pieces of training, the accuracy rate was stable at around 94.01\%.

Given a sentence S containing noun n_I, the similarity measure of strings is expressed as:

$$\frac{1}{N}\sum_{n_i}^{n_n} m_i \cdot \cos(n_i, c)$$

The metric Eqa. 12 generates a comparison between the dataset and the generated sentence every five training batches to measure the semantic proximity between the context words provided and the generated text. The cosine similarity is depicted in Fig. 8.

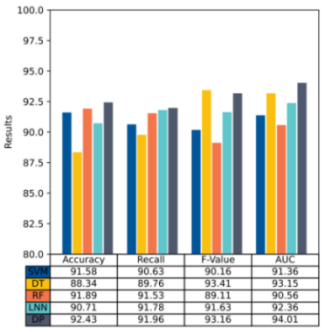

Figure 7: Activity status evaluation result.

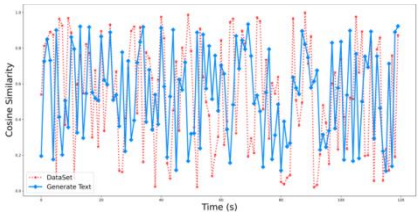

Figure 8: Cosine similarity.

IV. CONCLUSION

In this work, we present **ECHOES**, a side-channel analysis framework that exploits motion sensor data and shared memory fluctuations to infer user input on touchscreen keyboards. By combining convolutional and recurrent neural networks with dynamic temporal regularization, ECHOES accurately detects motion states and click regions during text entry. To alleviate the inherent ambiguity in precise key localization, we propose a region-based keyboard partitioning strategy and employ natural language processing techniques to recover meaningful sentence-level input. Experimental evaluations demonstrate that our approach achieves high recognition accuracy in both motion state classification (94.01%) and input restoration (91.63%), highlighting its effectiveness in recovering user input and exposing potential privacy risks. This study not only underscores the feasibility of shared memory as a viable side channel but also reveals the security implications of increasingly sensor-rich mobile devices. In future work, we aim to enhance input recovery granularity, extend the attack model to more complex typing behaviors, and explore potential countermeasures to mitigate such threats.

REFERENCES

[1] M. J. Hossain, B. T. Tabatabaei, M. Kiki, and J.-W. Choi, "Additive manufacturing of sensors: A comprehensive review," International Journal of Precision Engineering and Manufacturing-Green Technology, vol. 12, no. 1, pp. 277–300, 2025.

[2] W. Ju, Y. Gu, Z. Mao, Z. Qiao, Y. Qin, X. Luo, H. Xiong, and M. Zhang, "Gps: Graph contrastive learning via multi-scale augmented views from adversarial pooling," Science China Information Sciences, vol. 68, no. 1, p. 112101, 2025.

[3] L. Gong, P. He, W. Yichen, D. Zhaoyang, and C. Zhu, "Vibration control of full rotational speed range for magnetically levitated high-speed motors based on linear quadratic regulator," Journal of Vibration Engineering & Technologies, vol. 13, no. 1, pp. 1–15, 2025.

[4] R. Xiong, X. Xu, Y. Liu, S. Du, L. Jin, F. Chen, and T. Wu, "A miniaturized mems accelerometer with anti-spring mechanism for enhancing sensitivity," Microsystems & Nanoengineering, vol. 11, no. 1, p. 42, 2025.

[5] A. R. Javed, M. O. Beg, M. Asim, T. Baker, and A. H. Al-Bayatti, "Alphalogger: Detecting motion-based side-channel attack using smartphone keystrokes," Journal of Ambient Intelligence and Humanized Computing,pp. 1–14, 2020.

[6] L. Cai and H. Chen, "{TouchLogger}: Inferring keystrokes on touch screen from smartphone motion," in 6th USENIX Workshop on Hot Top- ics in Security (HotSec 11), 2011.

[7] Y. Deng and Y. Zhong, "Keystroke dynamics user authentication based on gaussian mixture model and deep belief nets," International Scholarly Research Notices, vol. 2013, no. 1, p. 565183, 2013.

[8] K. Zaman, M. Sah, C. Direkoglu, and M. Unoki, "A survey of audio classification using deep learning," IEEE Access, vol. 11, pp. 106 620–106 649, 2023.

[9] Lukasz Wycislik, Przemyslaw Wylezek, and Alina Momot. The improved biometric identification of keystroke dynamics based on deep learning approaches. Sensors, 24(12):3763, 2024.

[10] S. V. Adve and K. Gharachorloo, "Shared memory consistency models: A tutorial," computer, vol. 29, no. 12, pp. 66–76, 1996.

[11] Y. Sun and Y. Sun, "An optimized tv regularization algorithm for image reconstruction of co-planar array capacitive sensor," IEEE Transactions on Instrumentation and Measurement, 2025.

[12] A. Joshi, R. Dabre, D. Kanojia, Z. Li, H. Zhan, G. Haffari, and D. Dippold, "Natural language processing for dialects of a language: A survey," ACM Computing Surveys, vol. 57, no. 6, pp. 1–37, 2025.

[13] R. C. Pena, "Next steps in 'bringing upstairs care downstairs'," Emergency Medicine News, vol. 45, no. 4, pp. 3–4, 2023.

[14] S. Azenkot and S. Zhai, "Touch behavior with different postures on soft smartphone keyboards," in Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services, 2012, pp. 251–260.

[15] X. Zhang and W. Cao, "Research on time series forecasting method based on autoregressive integrated moving average model with zonotopic kalman filter." Sustainability (2071-1050), vol. 17, no. 7, 2025.

[16] J. Liang and X. Mao, "Rectifier fault diagnosis based on euclidean norm fusion multi-frequency bands and multi-scale permutation entropy," Electronics, vol. 14, no. 3, p. 612, 2025.

[17] D. Iacobucci, S. Román, S. Moon, and D. Rouziès, "A tutorial on what to do with skewness, kurtosis, and outliers: New insights to help scholars conduct and defend their research," Psychology & Marketing, 2025.

[18] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101, 2017.

[20] X. Xiang, X. Ding, L. Jin, Z. Li, J. Tang, and R. Jain, "Alleviating overfitting in hashing-based fine-grained image retrieval: From causal feature learning to binary-injected hash learning," IEEE Transactions on Multimedia, 2024.

[21] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large language models: A survey," arXiv preprint arXiv:2402.06196, 2024.

[22] Z. Xue, G. Yang, X. Yu, A. Yu, Y. Guo, B. Liu, and J. Zhou, "Multimodal self-supervised learning for remote sensing data land cover classification," Pattern Recognition, vol. 157, p. 110959, 2025.

[23] Jane Oruh, Serestina Viriri, and Adekanmi Adegun. Long short-term memory recurrent neural network for automatic speech recognition. IEEE Access, 10:30069–30079, 2022.

[24] V. Bertalan and D. Aloise, "Clustering textual features for log summarization in large software systems," 2025.

[25] Atin Roy and Subrata Chakraborty. Support vector machine in structural reliability analysis: A review. Reliability Engineering & System Safety, 233:109126, 2023.

[26] Vinícius G Costa and Carlos E Pedreira. Recent advances in decision trees: An updated survey. Artificial Intelligence Review, 56(5):4765-4800, 2023.