# Deep Learning Based Inference of Private Information Using Embedded Sensors in Smart Devices

Yi Liang, Zhipeng Cai, Jiguo Yu, Qilong Han, and Yingshu Li

## Abstract

Smart mobile devices and mobile apps have been rolling out at swift speeds over the last decade, turning these devices into convenient and general-purpose computing platforms. Sensory data from smart devices are important resources to nourish mobile services, and they are regarded as innocuous information that can be obtained without user permissions. In this article, we show that this seemingly innocuous information could cause serious privacy issues. First, we demonstrate that users' tap positions on the screens of smart devices can be identified based on sensory data by employing some deep learning techniques. Second, it is shown that tap stream profiles for each type of apps can be collected, so that a user's app usage habit can be accurately inferred. In our experiments, the sensory data and mobile app usage information of 102 volunteers are collected. The experiment results demonstrate that the prediction accuracy of tap position inference can be at least 90 percent by utilizing convolutional neural networks. Furthermore, based on the inferred tap position information, users' app usage habits and passwords may be inferred with high accuracy.

## Introduction

The usage of smart mobile devices for personal and business purposes has grown increasingly in popularity over the last decade. Unfortunately, mobile attacks have simultaneously exploded and become more sophisticated, especially when more and more users rely on mobile devices to manage their financial and personal data. Many services in smart devices are reliant on the sensory data collected from the embedded sensors. Those sensory data nourish the mobile app design to provide incredibly convenient services to people. In most mobile platforms, the sensory data readings are considered non-sensitive and can be easily collected without user permission. Recent studies indicate that freely accessible built-in sensors can be easily utilized by adversaries to launch inference attacks [1–4]. Some recent works mentioned that built-in sensors can be utilized to recognize human activities [5] and infer screen based input [2] based on the assumption that different human activities or gestures can create unique sensory data "patterns."

In this article, we propose a novel approach to identify user tap position even if we do not have any historical sensory data of this particular user. Furthermore, we make use of the identified tap patterns for each type of apps to further discover users' app usage habits and daily life patterns, which are definitely private information to most users.

Intuitively, different types of apps incur different tap patterns due to the usage nature of each app. For instance, users type frequently and fast in chatting apps such as Snapchat and WeChat, while users scroll down/up on screens and type occasionally when reading news. Based on those observations, we first investigate how to infer tap positions based on sensory data, the inferred tap positions can then help with deriving tap sequences, according to which we can record the traces of tap events of users when they use smart devices. A tap sequence consists of a series of positions tapped by a user on a screen. Tap sequences can help with distinguishing different apps and predicting apps being used or used before, which is referred to as a usage habit of a user. Note that usage habit information is private and can be used by adversaries to infer more private information such as age, gender, etc.

In order to achieve the aforementioned goals, we propose several models to encode tap sequences and the intervals between taps in a tap sequence. Specifically, *n*-gram is used to measure the similarity distance between two tap sequences, and a few machine learning models are applied to recognize the app being used by a user. We validate our work using the sensory data collected from 102 volunteers. The experimental results demonstrate that our proposed deep learning based method can predict tap sequences and infer app usage habits with high accuracy. The key contributions of our work are summarized as follows:

1. To the best of our knowledge, this is the first work to demonstrate that tap sequence can distinguish apps accurately, which reveals the fact that seemingly innocuous sensory data from smart devices can seriously threaten user privacy.
2. Several methods and models are proposed and evaluated in this article. We employ some traditional classification methods as well as deep neural network to infer tap positions. Our experiment results demonstrate that deep learning methods, such as Convolutional Neural Network (CNN), is very effective for inferring tap sequences. Furthermore, we propose a robust model to infer app usage habits of a user.

Digital Object Identifier:
10.1109/MNET.2018.1700349

*Yi Liang, Zhipeng Cai, Yingshu Li are with Georgia State Univerisity; Jiguo Yu is with Qufu Normal University; Qilong Han is with Harbin Engineering University.*

3. All the experimental data are collected from real traces. We develop a new app on the Android system and run it on 102 volunteer smartphones. We "steal" data from their smart phones in a nontrivial way when they use their smartphones, and all the experiment results validate our hypothesis that sensory data can be easily utilized to carry out privacy attacks.

The rest of the article is organized as follows. We introduce the background knowledge, followed by tap position inference. We study how to infer some private information based on inferred tap positions. Our experiment results are presented, and we introduce future research directions. Finally, we conclude the article.

## Background Knowledge

### Sensors in Smart Devices

With the emergence of smart devices with touch screens, users rely more and more on smart devices to deal with daily business, even for extremely private and sensitive business involving personal and financial data. In this article, we use smartphones as a case study. Nowadays, off-the-shelf smartphones are equipped with sensors that can provide various interaction functionalities. The most common sensors include accelerometer, gyroscope, and rotation vector. Each kind of sensors can sense in three dimensions. We denote the nine dimensions as $A_x$, $A_y$, $A_z$, $G_x$, $G_y$, $G_z$, $R_x$, $R_y$, and $R_z$, respectively. Tap actions on the screen of a smartphone can be easily captured by these sensors.

### Correlation Between Sensory Data and Tap Positions

It is well known that malicious apps can steal sensory data secretly because these sensors can be accessed without user permission [6], resulting in serious privacy issues. In this article, we first show that what we type through the soft keyboard, which is the most common input method in smartphones, can be inferred through sensory data. Note that very sensitive information such as passwords, PINs, social security numbers, and credit card numbers are generally input through the soft keyboard. Then, we demonstrate that current running apps can also be identified by mining sensory data. The information regarding running apps is also sensitive. Unfortunately, most users are not even aware of this fact.

It is not surprising that side channel information can be utilized by attackers [2, 7, 8]. Many previous works have shown that the changing angle and vibration of a touch screen on a smartphone are highly correlated to tap positions. In Fig. 1, the lines with different colors represent multiple taps on the same position. Each sub-figure depicts a specific dimension of a sensor. Similarity presents in each of the nine dimensions of sensory data. Then we derive the observation that a tap position has unique sensory data patterns. Thus we can infer tap positions according to sensory data. Our experiment results show that we are able to infer tap positions with 99 percent accuracy.

Furthermore, we observe that same kind of apps share similar user interface layouts, e.g., chatting apps such as *WeChat* and *Messenger*. Users carry out similar actions for the same kind of apps. We believe that the similarity of the same kind of apps is unique and can be used to distinguish different kinds of apps. Our experiment results also validate this fact.

To infer the app usage of a user, we can record all the sensory data when the user uses an app. Based on our tap position inference results, we can derive a tap sequence representing a unique pattern for each kind of app. As long as we collect enough training data and tune our inference model well enough, this inference model can be used to infer an app.

### Attack Model

In our attack model, we assume users have been tricked to install our malicious app on their smartphones so that we can collect their sensory data [9]. The most common way is to develop an app similar to a popular paid app and make it free in the Android app store. Lots of careless users will be tricked. Once a user launches the malicious app, the app starts to collect sensory data secretly. Then the malicious app can send the sensory data back to our back-end to train our inference models and launch inference attacks.

Our system consists of several components including tap detection, keystroke recognition, tap position inference, tap sequence pattern recognition, and app inference, as shown in Fig. 2. Initially, a tap event is captured by the tap detection component when a user taps on the screen. Then many features can be extracted from the sensory data and easily associated with tap positions during our training process. If a user switches to another app, we can detect a tap event and record sensory data. The sensory data will be compared with our training data to infer tap positions.

To infer the app usage of a user, we can record all the sensory data when the user uses an app. Based on our tap position inference results, we can derive a tap sequence representing a unique pattern for each kind of app. As long as we collect enough training data and tune our inference model well enough, this inference model can be used to infer an app.

### Sensory Data Collection

To collect sensory data from smartphones secretly, we design and implement a trojan app, named *Informer*, on the Android platform which has two parts, *sensor reading service* and *host app*. The sensor reading service is responsible for gathering sensory data from smartphones. The host app is a luringly installed malicious app such as tools, media, and games. We can "steal" sensory data from users' smartphones without users' notice because the sensors in smartphones can be accessed without user permissions [6]. *Informer* has two stages: training data collection and sensory data recording. In the training data collection stage, users interact with the host app so that the tap positions and sensory data can both be collected. Then we can extract the features of the sensory data for different positions. In this way, we can easily associate sensory data with tap positions to form an inference database. In the sensory data recording stage, if a user is not interacting with the host app, *Informer* cannot capture tap events and positions. However, we can still infer

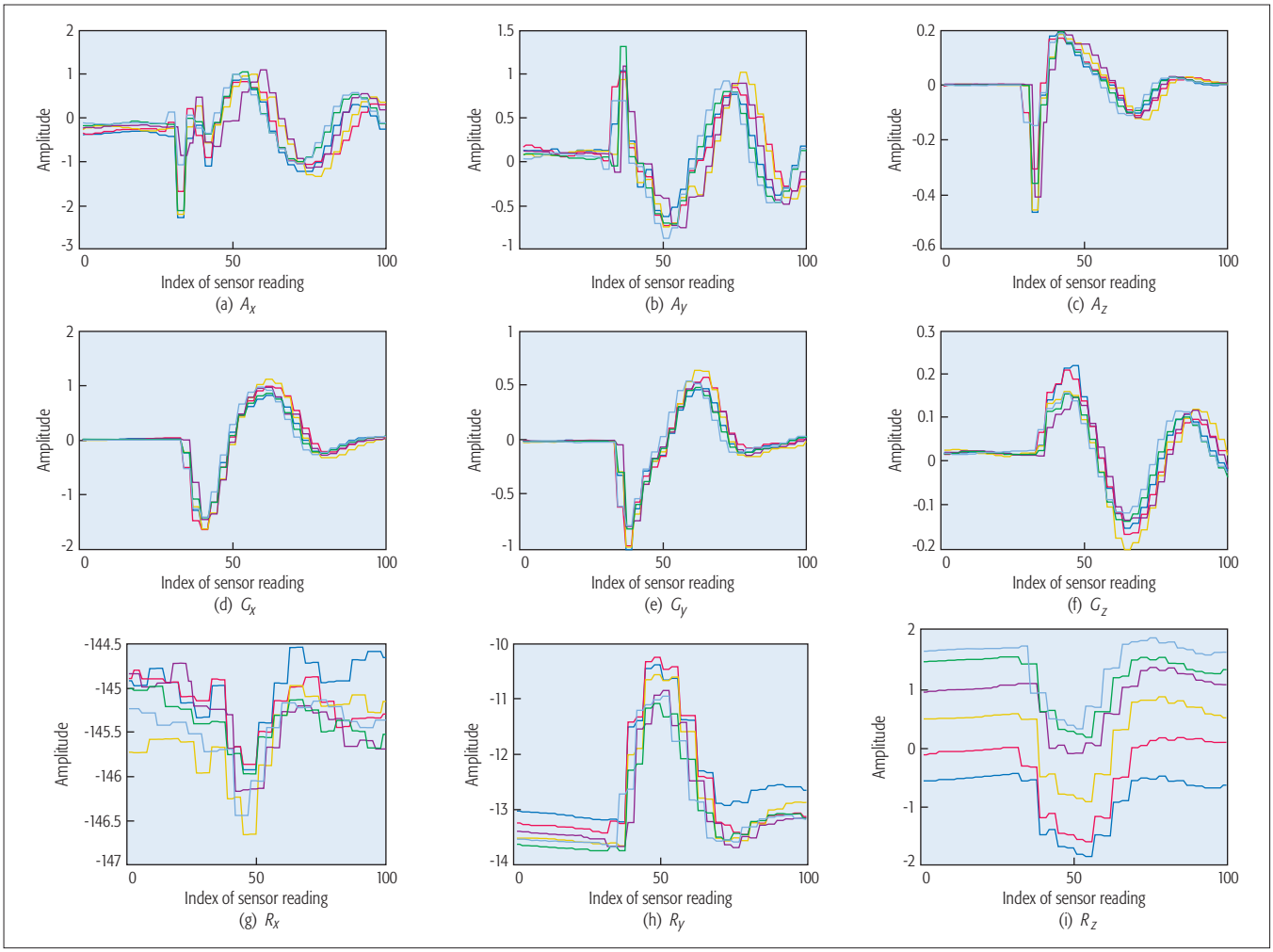


FIGURE 1. Similarity of sensory data for a same tap position.

tap positions. Because the sensor reading service keeps recording sensory data which can be used for inferring tap positions. In order to collect sensory data for all the possible positions on a screen, we design the layout of the host app carefully so that a user has to tap all the possible positions on a screen when interacting with *Informer*.

## Traditional Methods

Many approaches have been proposed for tap detection and recognition [1, 2, 7]. However, our experiment results show that the previous methods have some limitations. Initially, we believe that sensory data should show very different patterns for different tap positions in each dimension. However, this is not always true. For instance, no matter where you tap on the screen, there must be a downward power impact on the smartphone. So the sensory data pattern is very similar for the $A_z$ axis regardless of tap positions. It is then impossible to distinguish tap positions simply based on the extracted features from an individual dimension. Thus, the methods in [7] and [2] may not be effective.

We also find the correlation among axes is unique for each tap position and is very stable, as shown in Fig. 3, between the angle of roll and pitch, which depicts the angle relation between roll and pitch for different tap positions. It is shown in each subfigure that different tap actions at the same position result in highly similar correlations among different types of sensory data. Therefore, we consider not only the features of each type of sensor data, but also the correlations among different types of sensory data.

**Tap Event Detection:** In the training data collection stage, our data collection app naturally receives tap events. In the sensory data recording stage, we can only derive tap events through sensory data. Thus, in this stage, the main challenge is to detect tap events, for which we only take accelerometer into consideration. In our experiments, we find that no matter where you tap on the screen, there is a great impact on the accelerometer along axis $A_z$. It is intuitive because all tap actions result in a downward power on the screen. Hence, we mainly utilize the sensory data of $A_z$ to detect tap actions. We first normalize raw sensory data, then set a threshold $\lambda$ for the square sum $SquareSum = (A_z^i)^2$. If the square sum exceeds $\lambda$, there is a peak candidate at time $i$. We may obtain many peak candidates and we need to filter out noises. There must be an interval between two sequential tap actions, and the peak width should fall into a constant range. So we set another four thresholds for the peaks in $A_z$, which are the minimum peak interval length, minimum peak height, minimum peak width, and maximum peak width. Then, all tap actions can be captured from the sensory data.

**Feature Extraction:** We can obtain an array of peak indices from the tap detection module. In our experiments, we use these peaks as an approximate index and cut off the small sections before and after these peak indices. Let us call these sections tap event windows which are processed respectively. It means we extract features for each axis respectively so that we can combine all the features of the nine dimensions of sensory data. The extracted features include the min, max, average, number of peaks and crests, and the index difference between min and max.

As previously mentioned, we not only extract features for each axis respectively, but also take the correlations among axes into consideration. We extract a total of 136 features for each tap action. It is very time-consuming and unnecessary to leverage all these 136 features to discriminate different tap positions, because we do not know which features contribute more to the characteristics of a tap action's sensory data for different positions. If we can recognize the features that have the strongest correlations with a tap position, we can not only reduce noise but also improve inference speed. We utilize principle component analysis (PCA) to filter features. In our experiments, after applying PCA, 10 features result in almost the highest accuracy.

**Tap Classification:** To infer a tap position, we adopt three methods, i.e., *k*-nearest neighbor (KNN), decision tree, and SVM, to perform classification. In KNN, for each tap action, we calculate the standard Euclidean distance between this tap and all the taps in the training data set under the new coordinate system. We check the majority ones among the closest five taps, then label this tap with the majority tap position.

### Convolutional Neural Networks

Traditional tap inference methods have some non-negligible drawbacks, e.g., feature extraction sacrifices data information. We propose to employ Convolutional Neural Networks (CNN) to infer tap positions. CNN is one of the most popular deep learning methods and has attracted much attention [10]. In particular, CNN has become a powerful tool in many areas, especially in image recognition and natural language processing [11, 12]. CNN is a feed-forward artificial neural network that consists of convolutional layers, pooling layers and fully connected layers. The convolutional layer and pooling layer can be viewed as a whole and stacked together so that we can create a CNN model as complex as possible. CNN requires that the input data have sort of "spatial" correlations, such as image data and digital signal data [13, 14]. In our experiments, we observe that the unique patterns of tap actions are described by the "shape" of the curve. The "shape" of the sensory data curve shown in Fig. 1 can be considered as signal data that have spatial correlation. Inspired by multi-channel image processing, our sensory data can be naturally treated as multi-channel images as there are nine axes of sensory data for each tap action. Different from image processing, where the input is a 2-dimensional array, our input is just a 1-dimensional vector. It does not become more challenging to adopt CNN since we just need to adjust the kernel shape accordingly. In our CNN based model, the input data is 81 by 1, the kernel size is 15 by 1, and 60 kernels are used in each layer. The stride is set to 1.

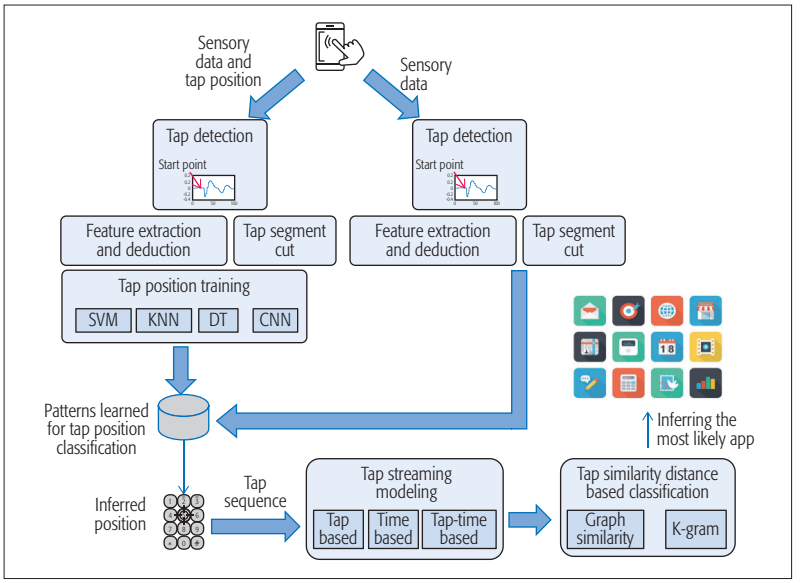


FIGURE 2. System overview.

There are many attractive advantages in using CNN to address the classification problem. One of the most powerful strengths of CNN is that we do not have to extract the features of tap sensory data manually. Inappropriate feature extraction leads to catastrophic consequence for classification. Even an experienced data analyst can hardly guarantee the effectiveness of feature extraction. While in CNN, all the important features are "extracted" automatically during the weight updating back-propagation process. The "spatial" correlations are also accommodated through parameter sharing. These superior strengths make CNN very suitable for solving our classification problem. This conclusion is also validated by our experiment results, which show CNN significantly outperforms the traditional methods.

## Applications of Tap Inference

Now we introduce two possible applications of tap inference.

### App Usage Inference

A tap position can be inferred accurately as discussed earlier. So we can infer app usage based on the obtained tap sequences. Intuitively, apps with similar functions should have similar operation patterns. For example, in social media apps, we can chat with friends and browse content shared by friends. In news apps, we keep scanning news until we find something attractive, then we click the news link to read it carefully.

In this article, we explore the feasibility of inferring app usage. A tap sequence refers to a series of tap actions. First, we explain how to model tap sequences and measure the similarity of tap sequences.

**Tap Sequence Modeling:** In our experiments, we divide a smart device's screen into nine zones similar to a numeric-only keyboard on a smartphone, defined as $TL = \{l_1, l_2, l_3 \cdots l_9\}$. We record both the timestamp and tap position for each tap event. Let $t_i$ be the timestamp for tap action $T_i$.

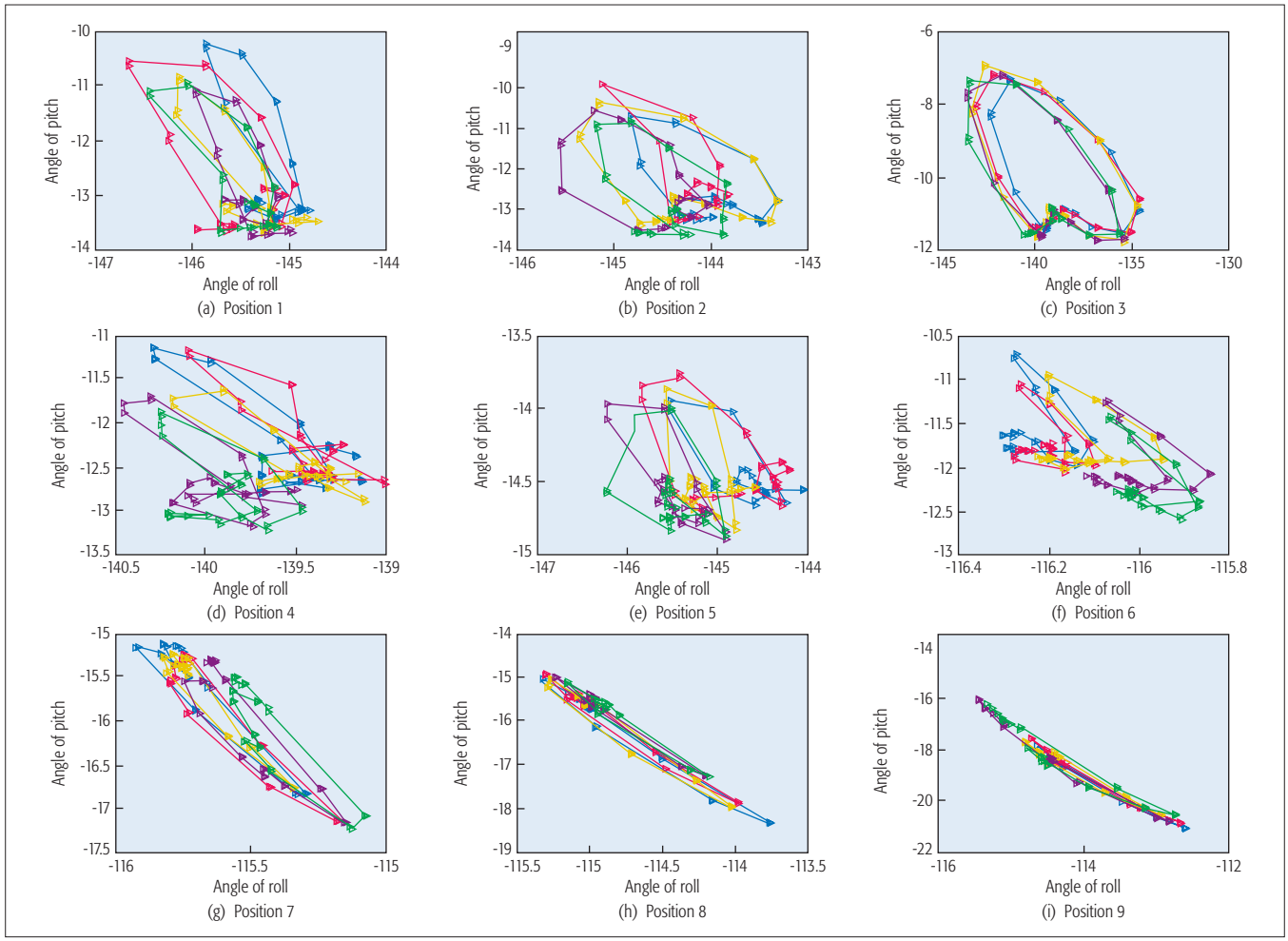


FIGURE 3. Correlation between angle of roll and pitch.

The following three models are considered for tap sequences.

***Position Based Model:*** The most straightforward approach is to only take tap positions into consideration. A tap sequence is recorded as a series of tap positions sorted by timestamps, e.g., $\{l_3, l_4, \cdots l_i, \cdots, l_8\}$ where $l_i \in TL$.

***Time Based Model:*** Our observation indicates that the interval between a pair of consecutive tap actions is an important factor to infer app usage. We model a tap sequence as a time interval sequence $\{t_2 - t_1, t_3 - t_2, \ldots t_n - t_{n-1}\}$, where $t_i - t_{i-1}$ is the time interval between tap action $T_i$ and tap action $T_{i-1}$. In order to utilize the *n*-gram algorithm, we categorize intervals into five groups $A$ = [0, 500ms), $B$ = [500ms, 1000ms), $C$ = [1000ms, 1500ms), $D$ = [1500ms, 2000ms), and $E$ = [2000ms, 2500ms). A time based sequence is then represented by a sequence of time intervals, such as $\{A, B, D\}$.

***Hybrid Model:*** We take both tap positions and time intervals into consideration. We model a tap sequence as a list of tap positions and time intervals. For example, $\{l_3, A, l_1, B, l_4, A, l_5, C \ldots l_9, D\}$.

We group the tap sequences of the same type of apps together to form a "profile" and utilize it to determine to which type of apps the upcoming tap sequence belongs.

**Tap Sequence Similarity:** Now we discuss the metrics for measuring the similarity between two tap sequences. We mainly focus on *n*-gram similarity and average element-wise matrix similarity [15].

***n*-gram similarity.** The ratio of the common subsequences of two tap sequences over the total number of subsequences can be utilized to measure their similarity [15]. Let $S_n$ be a set of subsequences of length $n$ appearing in one tap sequence. Thus, for tap sequence $Q = \{q_1, q_2, \ldots, q_N\}$, $S_k(Q) = \{subseq \mid subseq = \{q_i, q_{i+1}, \cdots, q_{i+n-1}\}, i \in [1, N + 1 - n]\}$. The similarity of two sequences $Q_1$ and $Q_2$ is defined as

$$Distance(Q_1, Q_2) = 1 - \frac{|S_n(Q_1) \cap S_n(Q_2)|}{|S_n(Q_1) \cup S_n(Q_2)|}. \quad (1)$$

*n*-gram follows the assumption that two similar apps should result in more common subsequences. The value of $n$ is the number of the necessary tap actions to complete an operation, and $n$ varies for different apps.

**Average element-wise matrix similarity.** We construct a transition matrix based on a tap sequence, where each node represents a tap position. There are two ways to define the weight of an edge from node $l_i$ to node $l_j$. The first way is to use the number of transitions from tap positions $l_i$ to $l_j$ over the total number of transitions. The other approach is to use the average transition time from tap positions $l_i$ to $l_j$. The similarity between two tap sequences is defined as the

average element-wise distance of their transition matrices. For each type of app, we collect a large number of tap sequences as training data. The similarity between the tap sequences in the training data set and a coming tap sequence is utilized to do the classification.

### Password Inference

Another important and more straightforward application of tap inference is password inference. We adopt a CNN based method for password inference where CNN outputs a probability vector for each single tap action. Each entry in the vector represents the probability of this position being tapped. We choose the tap sequence with the highest joint probability as our inferred password. As users are usually allowed to try three times when typing in a password, our method outputs the top three inferred passwords with the highest joint probabilities. If the real password is one of these top three ones, we consider the inference as correct.

## Experiment Results

We implemented our adversary app in an Android system API level 23 and tested it with many kinds of smartphones, including Samsung Galaxy S7 edge, Samsung Note 5, HUAWEI meta 9, HUAWEI Honor 8, LG G5, HTC M8, etc. Our volunteers held the smartphones with their left hand and used their pointer fingers to tap on the screens.

### Tap Inference

We had 102 volunteers and the sensory data for about 91,800 tap actions were collected. A smartphone screen is divided into nine zones. During the training data collection stage, each volunteer taps in each zone on the screen for at least 100 times. During the data recording stage, the volunteers use smartphones as usual. The average tap inference accuracies are 62 percent, 38 percent, 69 percent, and 93 percent for KNN, Decision Tree, SVM and CNN, respectively. For our CNN based method, more than 95 percent of inference accuracies are better than 80 percent, which is very impressive.

The traditional methods mainly focus on feature extraction, which might sacrifice accuracy. As shown in Fig. 4, for SVM and KNN, with only 10 features, we can have an accuracy of 80 percent. As the number of features increases, the accuracy is not substantially improved. This validates that feature extraction based methods may not be effective in practice.

### App Inference

For app inference, we classify the apps into seven categories based on their functions, as shown in Table 1.

We employ three tap sequence models as introduced earlier. In the position based model (Tap) and time based model (Interval), we use *n*-gram distance to measure app similarity distance. For the hybrid model, we adopt three similarity distance methods: *n*-gram (Hybrid), graph-count (GC), and graph-interval (GT). As introduced earlier, we build a transition "graph" based on these nine tap positions. GC uses the number of average tap transition counts between different tap positions as the edge weight, while GT utilizes the average transition time interval between different tap positions as the edge weight. The results are shown in Fig. 5. Our hybrid model with *n*-gram (Hybrid) achieves the best results.

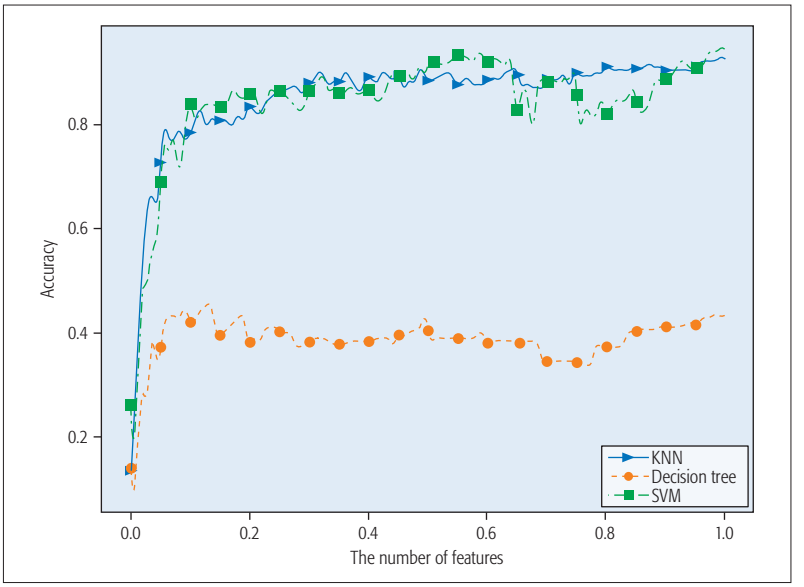


FIGURE 4. Impact of features.

| Categories | Apps |
|---|---|
| Game (TR) | Temple Run, Paris Metro |
| Shopping (SP) | TaoBao, Amazon |
| Video (VD) | Youku, Iqiyi |
| Browser (BR) | QQ browser, UC browser |
| Social media (SM) | Weibo, Facebook |
| Instant chat (IC) | WeChat, Messenger |
| Music player (MC) | QQ Music, WangYi Yun Music |

TABLE 1. Apps in our experiments.

### Password Inference

Although it is also related to tap positions, password inference is harder because it is considered as failed even with one single inference error. In our experiments, we consider passwords with different lengths, including 638 4-digit passwords, 529 6-digit passwords, and 336 8-digit passwords. If our top three candidates contain the correct password, we consider it as correct. For 4-digit passwords, the accuracy reaches 94.3 percent. For 6-digit and 8-digit passwords, the accuracies are 92 percent and 89.9 percent, respectively.

## Future Research Directions

**Detection and prevention of sensor-based attacks.** Attacks using sensory data are attracting increasing attention. Smart devices embedded with sensors provide attackers with opportunities to perform inference attacks. On the other hand, sensors can also be utilized to prevent and detect such kind of attacks. For instance, based on the battery consumption level, we can detect abnormal sensory data collection activities since such activities consume lots of power.

**Deep learning-based inference.** Deep learning approaches have been proven successful in many areas. However, they have not been widely adopted to prevent inference attacks. In our CNN based model, we just employ a two-layer convolutional and pooling network, which is very simple

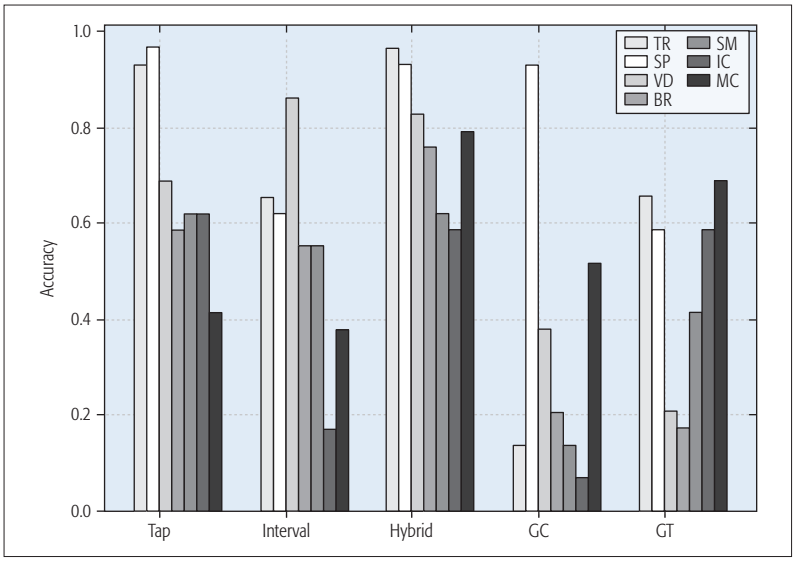


FIGURE 5. App inference accuracy.

and basic. A more complex CNN based model or novel deep learning methods need to be developed to seek improved and stable results.

**Fusion of multi-modal data.** Data fusion can help us derive more consistent and accurate results. However, current research only focuses on a specific type of sensory data. Smart devices are usually equipped with many different kinds of sensors, and user activities may be captured through various sensors. Therefore, how to fuse multi-modal data and make use of the fused data to prevent attacks are worth a thorough investigation.

## Conclusion

Though smart devices are indispensable in modern life, most people do not realize that smartphones also threaten their privacy. People usually ignore the fact that sensory data can be secretly collected from the sensors embedded in smart devices without user permission. In this article, we present the feasibility of inferring users' app usage habits solely based on sensory data. More specifically, we propose three improved traditional methods and one deep neural network method to infer users' tap positions by analyzing the secretly collected sensory data. The extensive experiment results show that our proposed methods achieve high accuracy and are very effective for tap classification, app inference and password inference.

## Acknowledgment

This work is partly supported by the National Science Foundation (NSF) under grant NO.CNS-1252292, 1704287, and 1741277; the NSF of China under contract 61373083, 61370084, 61502116, 61371185 and 61373027; the NSF of Shandong Province under contract ZR2012FM023; and the China Postdoctoral Science Foundation NO.2015M571231.

## References

[1] L. Cai and H. Chen, "Touchlogger: Inferring Keystrokes on Touch Screen from Smartphone Motion," *HotSec*, vol. 11, 2011, pp. 9–9.
[2] Z. Xu, K. Bai, and S. Zhu, "Taplogger: Inferring User Inputs on Smartphone Touchscreens Using On-Board Motion Sensors," *Proc. 5th ACM Conf. Security and Privacy in Wireless and Mobile Networks*, 2012, pp. 113–24.
[3] M. Goel, J. Wobbrock, and S. Patel, "Gripsense: Using Builtin Sensors to Detect Hand Posture and Pressure on Commodity Mobile Phones," *Proc. 25th Annual ACM Symp. User Interface Software and Technology*, 2012, pp. 545–54.
[4] L. Pei *et al.*, "Human Behavior Cognition Using Smartphone Sensors," *Sensors*, vol. 13, no. 2, 2013, pp. 1402–24.
[5] M. Fahim *et al.*, "Daily Life Activity Tracking Application for Smart Homes Using Android Smartphone," *2012 14th Int'l. Conf. IEEE Advanced Communication Technology (ICACT)*, 2012, pp. 241–45.
[6] Google, 2017, Android Developer, https: //developer.android.com/.
[7] A. J. Aviv *et al.*, "Practicality of Accelerometer Side Channels on Smartphones," *Proc. 28th Annual Computer Security Applications Conference*, 2012, pp. 41–50.
[8] E. Miluzzo *et al.*, "Tapprints: Your Finger Taps Have Fingerprints," *Proc. 10th Int'l. Conf. Mobile Systems, Applications, and Services*, 2012, pp. 323–36.
[9] X. Liu *et al.*, "When Good Becomes Evil: Keystroke Inference with Smartwatch," *Proc. 22nd ACM SIGSAC Conf. Computer and Communications Security*, 2015, pp. 1273–85.
[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, 2015, pp. 436–44.
[11] B. Shi, X. Bai, and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2016.
[12] L. Xu , "Deep Convolutional Neural Network for Image Deconvolution," *Advances in Neural Information Processing Systems*, 2014, pp. 1790–98.
[13] M. Zeng *et al.*, "Convolutional Neural Networks for Human Activity Recognition Using Mobile Sensors," *2014 6th Int'l. Conf. IEEE Mobile Computing, Applications and Services (MobiCASE)*, 2014, pp. 197–205.
[14] J. Yang *et al.*, "Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition," in IJCAI, 2015, pp. 3995–4001.
[15] G. Wang *et al.*, "You Are How You Click: Clickstream Analysis for Sybil Detection," *Usenix Security*, vol. 14, 2013.

## Biographies

Yi Liang (yliang5@gsu.edu) is currently a Ph.D. student in the Department of Computer Science at Georgia State University. He received his M.S. degree in computer science from the University of Science and Technology of China. His research interests include privacy preservation and social networking.

Zhipeng Cai (zcai@gsu.edu) is an associate professor in the Department of Computer Science at Georgia State University. He received his B.S. degree from Beijing Institute of Technology, and his M.S. and Ph.D. degrees in computing science from the University of Alberta. His research interests include big data, privacy aware computing, wireless networks, and optimization theory. He is an associate editor for *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Vehicular Technology*, and *IEEE Access*, among others. He is the recipient of the NSF CAREER Award.

Jiguo Yu received the Ph.D. degree in the School of Mathematics from Shandong University in 2004. Since 2007 he has been a professor in the School of Computer Science, Qufu Normal University, Shandong, China. He is currently a professor in the School of Information Science and Engineering, Qufu Normal University. His main research interests include wireless networks, distributed algorithms, peer-to-peer computing and graph theory. In particular, he is interested in designing and analyzing algorithms for many computationally difficult problems in networks. He is a senior member of the CCF (China Computer Federation).

Qilong Han received the Ph.D. degree in computer science from Harbin Institute University, Harbin, China, in 2006. He is currently a professor and Deputy Dean in the College of Computer Science and Technology, Harbin Engineering University. His research interests include data security and privacy, mobile computing, distributed and networked systems. He has more than 70 publications as edited books and proceedings, invited book chapters, and technical papers in refereed journals and conferences. He is a senior member of CCF, and the Chair of CCF YOCSEF Harbin.

Yingshu Li (yili@gsu.edu) received her Ph.D. and M.S. degrees from the Department of Computer Science and Engineering at the University of Minnesota-Twin Cities. She received her B.S. degree from the Department of Computer Science and Engineering at Beijing Institute of Technology, China. She is currently an associate professor in the Department of Computer Science at Georgia State University. Her research interests include wireless networking, sensor networks, sensory data management, social networks, and optimization. She is the recipient of an NSF CAREER Award.