DeepFP: A Deep Learning Framework For User Fingerprinting via Mobile Motion Sensors

Sara Amini*, Vahid Noroozi*, Sara Bahaadini†, Philip S. Yu*, Chris Kanich*
*Department of Computer Science, University of Illinois at Chicago, IL, USA
†Department of Electrical Engineering and Computer Science, Northwestern University, IL, USA
email:{samini3,vnoroo2}@uic.edu, sara.bahaadini@u.northwestern.edu, {psyu,ckanich}@uic.edu

***Abstract*—In this paper, we propose a deep learning framework for user fingerprinting via mobile motion sensors, DeepFP, which can identify and track users based on their behavioral patterns while interacting with the smartphone. Existing machine learning techniques for user identification are classification-oriented and thus are not amenable easily to large-scale, real world deployment. They need to be trained on all the users whom they want to identify. DeepFP exploits metric learning techniques and deep neural networks to address the challenges of current user identification techniques. We leverage feature embedding to directly extract informative features and map input samples to a discriminative lower-dimensional space, where recurrent neural networks are used to model the temporal information of data. DeepFP does not need to re-train to identify new users which makes it feasible to be used in real world scenarios with a huge number of users, without needing a large number of training samples. Experiments on a publicly available mobile sensors dataset and comparison with other embedding methods depict the effectiveness of DeepFP.**

***Index Terms*—deep learning, user fingerprinting, mobile motion sensors, user identification**

I. Introduction

Modern targeted advertising is predicated on the ability to perform that targeting: tracking users while they interact with content, applications, and other users online is the primary source of the information used for this targeting. While cookies are very often sufficient to complete this task, targeted advertising is such an immense economic engine that substantial effort has been made to fingerprint individual devices via unique properties which can often be used to identify individual web browsers [1]. However, alongside the proliferation of mobile devices, these approaches are not a silver bullet: while a large portion of users are uniquely identifiable via device fingerprints, an increasingly large portion are not: for instance, per Laperdrix et al., when flash is not available, 88% of browsers are uniquely identifiable, and 11.4% are in an equivalence set of at most 50, and the remaining .6% of users are in equivalence sets larger than 50. To successfully track a non-trivial and growing subset of these users, alternative approaches are necessary [2].

To close this final gap between a moderate sized equivalence class and individual identification, the best known methods for leveraging statistical inference techniques in this domain are largely based on classification oriented methods that have several drawbacks in this setting. Below we review such characteristics and challenges:

- Large number of classes: the absolute number of smartphone users is huge, but typical classifiers are not capable of handling the classification problem with huge number of classes/users. Due to their architecture, such approaches require re-training when new users enter the system. Thus, it is not feasible to use for a task such as online tracking both because of the volume and the velocity of user churn.
- Limited samples for each user: using machine learning methods as classifiers to identify different users requires sufficient data available in training phase. However, in the real world scenario a very few or even zero samples pertaining to individual users may be present.
- Unbalanced samples: the performance of these methods declines when it comes to dealing with unbalanced classes where the number of samples pertaining to different classes/users are not balanced.
- Privacy: since the training and updating processes of these methods are complex computationally, they cannot be performed on the device due to their limited storage and processing resources. Therefore, an additional service such as cloud computing is required to perform this calculation. Aggregating and storing raw, sensitive activity data for model training purposes is unpalatable for many companies, as it is seen as a risk and an undue invasion of user privacy.[1] Moreover, these methods require continuous network communication which may not be feasible for the identification and authentication tasks in real scenarios.

To address these issues, we propose a **Deep** learning framework for user **F**inger**P**rinting (DeepFP) which models users' behavioral motions from the sequential data captured from the motion sensors of their mobile phone. Our model learns to produce fingerprints for mobile users which can be used to identify and track them. It is performed by exploiting neural networks to learn a highly nonlinear feature embedding function to map samples of a user's mobile sensors data to a discriminative low-dimensional feature space where users are easily distinguishable. In the learned low-dimensional feature space, samples pertaining to the same user (matching

[1]For instance, Apple and Google are applying differential privacy preserving techniques in their collection of user data to limit the individually identifying data that leaves personal devices [3, 4].

samples) are close to each other, whereas those of different users (non-matching samples) are farther apart. In this manner, the classification problem is converted to a simple nearest neighbor problem, and a simple distance metric, such as Euclidean, can be used to find and identify a user among a large number of users.

The metric learning problem is commonly used in verification using linear or non-linear embedding [5, 6, 7]. In the context of our goal, verification aims to verify whether two samples pertain to the same user or not. The verification problem can be converted to an identification task by comparing the distance between a new incoming sample and those already stored in the history. Using the similarity metric learned by the model in the training phase, DeepFP is able to find the user with the closest samples to the new incoming sample. If none of samples in the history is close enough to the new incoming sample, it is labeled as a new user. Thus, in a system with millions or billions of users, re-training the model is not needed when users join or leave the system, or for the different equivalence classes, but rather only once at the outset of system deployment.

We leverage recurrent neural networks as the modeling part of our framework which can model highly non-linear relations and has showed significant results on modeling sequential data without any prior knowledge and domain expertise. Additionally, it helps the scalability of our model as neural networks are parametric models and their training complexity is linear in terms of the size of the training data.

We evaluate DeepFP on a publicly available mobile sensors dataset, H-MOG [8], and compare it to the linear and non-linear embedding baselines. As shown in the evaluation section later in this paper, DeepFP outperforms all the baselines in terms of accuracy.

II. Related Work

Most of the current related works to ours which use mobile sensors are user identification and continuous user authentication. Based on the user's distinguishable behavioral patterns inferred from sensors data [9], continuous user authentication aims to implicitly verify whether the current user is the actual legitimate user or not, whereas in the user identification problem the goal is to identify the current user of the mobile device correctly among all users. These works have used different modeling techniques from traditional machine learning methods such as Support Vector Machine (SVM) [10] and k-Nearest Neighbors (k-NN) [11] to deep learning techniques such as Convolutional Neural Networks (CNNs) [12], Long Short-Term Memory (LSTM) [13] and Recurrent Neural Networks (RNNs) [14]. In [15], Lee et al. adapted SVM to authenticate users implicitly and continuously based on three sensors: accelerometer, gyroscope, and magnetometer. Bo et al. [16] also used SVM and proposed SilentSense to authenticate users based on their touch behavior and the smartphone reaction (accelerometer and gyroscope sensors data). Lin et al. [17] deployed k-NN to perform a non-intrusive authentication method based on gyroscope sensor data. Amini et al. [18] uitilized LSTM to provide an implicit re-authentication mechanism that enables a frictionless and secure mobile user experience in the application via accelerometer and gyroscope motion sensors.

Moreover, there have been a number of studies on the user identification task, including IDNet [19] and GaitID [20]. In GaitID a model is proposed which is capable of extracting the gait template and identifying users using accelerometer sensor data and SVM as the classifier. IDNet is also a gait-based model that adapted CNNs to extract features and SVM to identify different users using accelerometer and gyroscope sensors data. Very similar to the approach used in this paper, DeepSense [21] uses time series mobile sensor data as input into an RNN. While the fundamental high level goal of the system is to identify individual users, this study was also done through a classification approach. Therefore it need model re-training for new users which may not be feasible for large number of users. Additionally, it needs the user's data to be sent to the server for model re-training which may violate users privacy.

A key difference between DeepFP and the aforementioned studies lies in the approach used to solve the problem. DeepFP proposes a solution to the identification problem through an embedding and fingerprinting approach, whereas all other works are classification-oriented models which are not feasible in real world scenarios and applications. Embedding learning based on neural networks has been used in various applications such as verification [5, 6, 7], classification for large number of classes [22], clustering [7], and etc.

To the best of our knowledge, we are the first to propose a deep model for modeling and tracking users from their mobile sensors which can be scalable and adoptable in real scenarios. Some characteristics of the user identification problem in the real world scenario are not well supported by the requirements of typical machine learning techniques used as classifiers. Unlike other works, DeepFP 1) is able to handle the huge number of users, 2) does not need to be trained for new users, 3) performs well when it encounters the problem of lack of sufficient information and data pertaining to users, 4) overcomes the challenge of imbalance classes, and 5) preserves privacy by not having the need to have access to the user's data directly. Since, it does not need to be trained for new users, it calculates the fingerprint of the user on the mobile device and just sends that to the server.

III. DeepFP Framework

A. Model Definition

Current mobile phones have various sensors including the accelerometer, digitizer (which captures touch events), gyroscope, and others. Most sensors capture the interaction and activity of mobile users and produce a series of samples over time, which can be used to model users individual behaviour. In this study, we are working with the sequential data obtained from users phones' sensors. For each user, we derive the individual's fingerprint using the sensor data gathered from her/his phone.

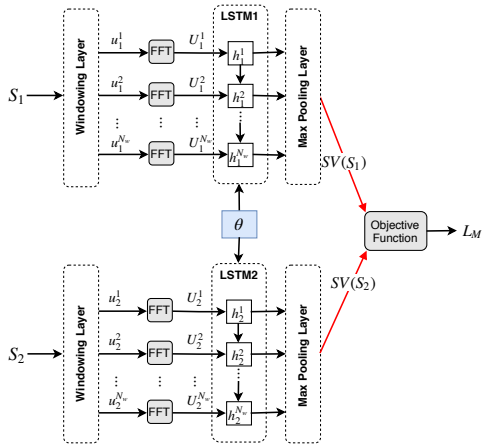


Fig. 1: Main architecture of DeepFP framework.

We define a segment of data as a sequence of measurements for T time steps. Each segment is denoted in the form of a matrix $S = [s_1, , \ldots, s_T]$, where vector $s_t \in \mathbb{R}^{D\times 1}$ represents the measurements at t^{th} time step, and D is the number of measurements.

Our aim is to learn an embedding function which gives the fingerprint of a user from its segment where fingerprints from the same user are close in the embedding space while fingerprints from different users are far from each other. We define such embedding function as $f_\theta(S) : \mathbb{R}^{D\times T} \rightarrow \mathbb{R}^{C\times 1}$ which maps the segment S of a user to the user's C dimensional fingerprint space. $f_\theta(.)$ is a nonlinear function parametrized by θ. A distance function $g_\theta(.,.)$ is also defined as

$$g_\theta\left(S, S'\right) = \left\| f_\theta(S) - f_\theta(S') \right\|_2 \tag{1}$$

which estimates the distance of two segments S and S' in the embedding space learned by f_θ. Conceptually, f_θ maps raw data from original feature space to another space (a lower dimensional one) in which Euclidean distance among data points equals to the distance between fingerprints.

We define the training samples in the form of pairs of segments. Some of the pairs have two segments from the same user which are labeled as *positive* pairs while others have segments from different users which are labeled as *negative* pairs. The model is trained in such way that positive pairs have a small distance in the new learned space and negative pairs are far from each other forming a discriminative feature space.

B. Model Architecture

The proposed model consists of six primary components: preprocessing, pairing, windowing, Discrete Fourier Transform (DFT), modeling, and finally an objective function. In the preprocessing layer, data gathered from different sensors are concatenated to form segments. In the pairing component, matching (positive) or non-matching (negative) pairs are made from all the available segments. The segments are divided into smaller units in the windowing layer, and each of such sub-segment is further mapped into the frequency domain by using Discrete Fourier Transform (DFT) [23] at the DFT layer. The modeling component is a neural network which learns a non-linear mapping function to map the processed segments to a new space where distances can be easily estimated by a simple Euclidean distance function. The final layer is the objective function that pushes the model to give the desired properties. In the following subsections, we discuss the details of these components. A schematic representation of the main architecture of the DeepFP framework starting from the windowing layer is shown in Figure 1.

1) Preprocessing: Most sensors are not usually synchronized with each other, their sampling is not performed uniformly, and it is done with unnecessary or inaccessibly high fidelity. Therefore we initially synchronize all of the sensor outputs and reduce their sampling rate. We then obtain uniformly sampled measurements over the same time intervals and thus with identical number of time steps for all sensors. We then concatenate all of the measurements from all sensors in each time step as the sample vector of that time step.

Assuming we have K sensors on the phone, and the k^{th} sensor at a time includes d_k different measurements (for example, the accelerometer sensor gives samples with three dimensions x, y and z which specifies the acceleration of the phone movement in three dimensional space), the total number of measurements in each time step is $D = \sum_{k=1}^{K} d_k$. As the samples for a user may be very long and also the length of input data for different users vary, we initially split the data of each user into fixed-size segments of size T time steps (see Section III-A). We perform the segmentation exploiting a windowing process by moving a fixed-size window of size T over the sequential data to build the segments. Thus for each user, we have a sequence of segments with size $D \times T$. The number of segments in each user's sequence depends on the length of the input data for that user. These segments are then passed to the pairing layer to build the training data.

2) Pairing: As mentioned in the previous subsection, after synchronizing, down-sampling, and segmenting the raw sensor data, we obtain a group of segments for each user. In this layer, we create the training pairs for each batch. As the model is based on neural networks, the training is preformed using batches of pairs. For the b^{th} batch, we create a set of training pairs $\mathcal{P}_b$ as

$$\mathcal{P}_b = \left\{ \left((S_1^i, S_2^i), y^i \right) \middle|\, y^i \in \{neg, pos\} \right\}_{i=1}^{N} \tag{2}$$

where (S_1^i, S_2^i) is a pair of segments $S_j^i \in \mathbb{R}^{D\times T}$, y^i is its corresponding label which indicates if the pair is from the same user *pos* or from different users *neg*. $N = N_n + N_p$ is the total number of training pairs including N_n negative pairs and N_p positive pairs for each batch.

The number of pairs that can be created from n segments is $O(n^2)$ which can be a large number, therefore we can not create all possible pairs. Moreover, if we create all possible pairs, the number of negative pairs will greatly outnumber positive pairs which can impair the learning process. To address such issues, we use a controlled random sampling to build the training sets for each batch. In the sampling process,

the ratio of negative to positive pairs is kept to a fixed number r, which we call the negative factor.

3) Windowing: The outputs of the pairing layer are then passed to the windowing layer. These segments, on their own, do not reveal distinctive behavioral patterns of users while using a mobile phone. Such patterns can be better detected if we look into a sequence of measurements instead of just exploring each single value, since one single measurement is rarely sufficient to reveal one's behavioral patterns while interacting with a mobile phone. These sequential measurements contain behavioral patterns that can be utilized to distinguish different users. In the windowing layer of DeepFP, we prepare segments to better reveal users' unique behavioral patterns while using a mobile phone. We categorize such patterns into micro and macro patterns. Micro patterns are those existing within a smaller portion of sequential measurements, whereas macro patterns are those involving a bigger sequence of measurements reflected by the inter-segment relationship of the micro patterns.

Preparation of segments for detection of micro and macro patterns are performed in windowing layer. We use the windowing technique to split segments into smaller windows to detect micro patterns. Let (S_1, S_2) be a pair of segments from the training set. We basically move a fixed-size window with size l over each segment with a pre-defined shift l_Δ to make smaller windows for each segment which we denote as: $Windowing(S_j) = \left\{u_j^i\right\}_{i=1}^{N_w}$, where S_j is a segment in the given pair, $N_w = \lfloor \frac{T}{l_\Delta} \rfloor$ is the number of windows in segment S_j, and $u_j^i \in \mathbb{R}^{D \times l}$ is the i^{th} window of the segment S_j.

Windowing is not only beneficial when it comes to detecting micro patterns, but also can reveal macro patterns and inter-segment relationships between subsequent windows. In the DeepFP model, the detection of macro patterns is performed in the recurrent layer. More details on this will be explained in Section III-B5.

4) DFT: Sensors' data are originally in the time domain. However, working in the frequency domain for our input data has the following advantages:

- **Noise handling and accuracy:** Measurements recorded from such devices usually contain a considerable amount of irrelevant and noisy signals that may result in accuracy degradation of behavioral pattern detection. Frequency domain data provides the ability to better handle and remove noise comparing to time domain.
- **Performance in detection of behavioral patterns:** Characteristics of the sensors' signals which are useful for distinguishing and detecting unique behavioral patterns can be better captured in the frequency domain.

Time domain signals can be converted to Frequency domain signals by computing the Discrete Fourier Transform (DFT). Fast Fourier Transform (FFT) is one of the most common and numerically efficient algorithms to estimate DFT. To transform windows of measurements in the time domain with size $D \times l$ to the frequency domain, we apply an FFT on each dimension of the window, i.e., a vector of size $1 \times l$ separately. As the FFT vectors are symmetric, we consider only half of them. The resulting FFT vectors are stacked on top of each other to form a feature vector with length of $\frac{l}{2} \times D$. This gives us a sequence of windows $\left\{U_j^i\right\}_{i=1}^{N_w}$ for segment S_j in the frequency domain, where U_j^i is the frequency form of window u_j^i. Putting all of these vectors into a two dimensional matrix gives S_j^f with dimension $(\frac{l}{2}D) \times N_w$, the frequency domain of segment S_j.

5) Modeling: The main goal of this study is to detect macro and micro patterns within mobile sensors data that can distinguish different users from each other. As mentioned earlier in this section, micro patterns are detected in preprocessing layer. By utilizing a recurrent layer in our model, we detect macro patterns which manifest as patterns occurring between subsequent windows.

Recurrent Neural Networks (RNNs) have appeared as one of the most popular neural networks for handling sequential data. To detect macro patterns, we use Long Short-Term Memory (LSTM) [13] which is a variant of RNNs with a better control over memory. It has been shown that LSTM outperforms other versions of RNNs when it comes to sequential data with larger time intervals due to its capability of learning long range dependencies through its use of memory cell units [24].

LSTM is employed to learn the non-linear embedding function $f_\theta(.)$ from a sequence of input vectors. θ here refers to the set of all parameters of the model. This maps the segments into a sequence of lower-dimensional feature vectors $[h_1, h_2, \ldots, h_{N_w}]$ where h_i correspond to the i^{th} input sequence and it has a pre-defined length of C. We use a max-pooling layer that chooses the maximum value from each dimension of the vectors $[h_1, h_2, \ldots, h_{N_w}]$ as the signature feature vector SV of the input segment S denoted as $SV = f_\theta(S)$.

6) Objective Function: The outputs of the recurrent layer (see Figure 1) are two C-dimensional signature feature vectors, $f_\theta(S_1) = SV_1$ and $f_\theta(S_2) = SV_2$. We define the *discriminative* difference between the two signature feature vectors of the i^{th} training pair by $g_\theta\left(S_1^i, S_2^i\right) = \left\|SV_1^i - SV_2^i\right\|_2$, using Euclidean distance. Subsequently, a marginalized contrastive loss function, L_M, shown in Equation 3 is employed that inspires the matching segments to be close together, whereas it reassures the non-matching segments to be far apart by enforcing a distance constraint. In other words, we aim to parametrize a non-linear mapping function f_θ by minimizing the contrastive loss function such that the distance $g_\theta\left(S_1^i, S_2^i\right)$ between S_1^i and S_2^i is smaller than a pre-specified threshold m if S_1^i and S_2^i segments are matching ($y^i = pos$), and larger than m if S_1^i and S_2^i are non-matching segments ($y^i = neg$). The loss function is of the form:

$$L_M = \sum_{i=1}^{N} y^i \left(g_\theta\left(S_1^i, S_2^i\right)\right)^2 + (1 - y^i)\left\{max(0, m - g_\theta\left(S_1^i, S_2^i\right))\right\}^2 \quad (3)$$

where N is the number of training pairs, $m > 0$ is a predefined margin and $y^i \in \{neg, pos\}$ demonstrate the labels of the i^{th} pair. The margin intensifies the distance constraint and prevents non-matching pairs that are beyond this margin from contributing to the loss.

C. Training DeepFP

The whole model is trained in an end-to-end manner using backpropagation with respect to the marginalized contrastive loss function illustrated in Equation 3. Given a set of N training pairs, we optimize the model's objective function using an adaptive version of the stochastic gradient descent method called Adam [25]. Moreover, we apply $l2$-regularization and dropout techniques to prevent overfitting.

IV. Identification Task

After minimizing the loss function 3 on the training users, we have learned the parameters θ of the function f_θ which can produce fingerprints for users. When we learn the function f_θ, the function g_θ can be used to estimate the distance between two segments from two users. If we could find a distance threshold τ such that $g_\theta(S_1, S_2) < \tau$ for matching segments and $g_\theta(S_1, S_2) > \tau$ for non-matching segments as much as possible, these vectors would function as reliable fingerprints. In other words, the threshold τ can distinguish pairs of matching fingerprints from pairs of non-matching fingerprints.

In our framework, users are divided into three non-overlapping sets: training, validation and testing sets. The overall procedure of using our framework is as follows. Initially, training users are used to learn the the parameters θ. Then, the validation set is used to estimate the most efficient value for the threshold τ. Consequently, our system is evaluated on the test users. We maintain a history of fingerprints from the users that we have observed or detected by the system. When a new user is detected, all samples from the user are split into segments and their corresponding fingerprints are generated by function f_θ. All of these fingerprints are added into a history of fingerprints. One of the main contributions of DeepFP is that it does not need to be trained on new users to be able to detect them, which is reflected in this evaluation framework. It can identify a user by just having at least one sample from the user in the history, and does not need to re-train the model when new users arrive.

In the identification phase of DeepFP a new segment S_{test} is identified either as an **Alien user** or a **Known user**. An *Alien user* is a user which is new to the system and it is the first observation of the user, whereas a *Known user* is a user for which we have some fingerprints in our history. Note that there is no overlapping between test and train users, and none of the users in the test are used in the training process. We apply the function g_θ to calculate the distances between the signature vector of segment S and all segments pertaining to user i. We denote the average of these distances (d_i) as the distance between the anonymous user and user i. We repeat this step for all users to get the distance between the anonymous user and all users in the data history. If all such distances are greater than the threshold τ, that means the anonymous user is an *Alien user*. On the other hand, if there are some users with a distance less than the threshold τ, that refers to the *Known user* scenario and the anonymous segment is pertaining to the one whose distance is the minimum among all users to the anonymous user.

V. Experiments

A. Dataset

A publicly available dataset of mobile sensors called H-MOG [8] is used in all of the following experiments. H-MOG was collected to explore behavioral biometric features for continuous user authentication [26]. It was collected from 100 volunteers while interacting with identical model Android phones via three tasks: reading, writing and navigation on a map. In total, each volunteer has completed 24 sessions including 8 sessions per task and they either sat or walked over the course of each session. In total, each volunteer contributed from 2 to 6 hours of interaction with mobile in the data collection. In H-MOG 9 different categories of data were collected including motion sensors, touch related data and keystroke data. We select accelerometer and gyroscope motion sensors data for our experiments. The sampling rate for accelerometer and gyroscope data stream is $100Hz$ in the H-MOG dataset. For the synchronizing and re-sampling step of DeepFP, we synchronize the data and down-sample the rate of these sensors data stream to $50Hz$.

B. Baselines

We evaluate DeepFP by comparing its effectiveness to other representation learning algorithms. Note that DeepFP cannot be compared with neural network based classification methods as they can not easily handle new classes. Traditional classification methods such as k-NN can support new classes but they are not able to extract efficient features as good as deep neural networks. Therefore, we adopt common embedding and representation learning methods along with two variants of the DeepFP model as our baselines. The baselines used in experiments are as follows: 1) Principal Component Analysis (PCA) [27], 2) Linear Discriminant Analysis (LDA) [28], 3) DeepFP(t) which is a variant of DeepFP which uses time domain features instead of frequency domain features, and 4) DeepFP(t+fft) which is another variant of DeepFP which adopts both time and frequency domains for features.

C. Experimental Settings

For the purpose of providing segments pertaining to users not present during model training, there is no overlap between users in training, validation and test sets. DeepFP is trained on 70% of the users, validated on 15% of them and tested on the remaining 15% of the users. After synchronizing and down sampling the accelerometer and gyroscope motion sensors data to the rate of $50Hz$, segmentation step is done over the sequential data to generate non-overlapping segments of size T. The negative factor $r = 5$ is used in the training process but

the ratio of positive and negative pairs in the validation and test process is considered as one. All segments are normalized to have zero mean with variance of one by calculating the mean and variance on the training data.

We use the default parameters of the Adam optimizer as provided in the original paper [25]. The batch size is 512 for all the training of neural networks. The size of the hidden states for LSTM, RNN, and GRU is set to 256. Dropout and $l2$-regularization parameters are set to 0.2 and $1\mathrm{e}{-4}$ respectively. Margin $m = 1$ is used in the loss function L_M. We experimented with different window sizes:$\{0.4, 0.8, 2, 4, 10\}$ seconds and selected $l = 4$ seconds which leads to the best performance. Window shift is set to $l_\Delta = \frac{l}{2}$. We tested with different values of the threshold τ for DeepFP and its variants from 0.2 to 0.8 and for PCA and LDA from 10 to 70. The best value is selected based on H accuracy measurement on the validation users. We experimented with different values of the C parameter (from 10 to 350) which is the size of the signature vectors and selected $C = 40$ which showed the best results in most scenarios on the validation set.

TABLE I: Performance of different methods on H-MOG in terms of accuracy for different segment sizes.

Segment Size	10 (s)			20 (s)			40 (s)		
Accuracy	N	P	H	N	P	H	N	P	H
DeepFP	**71.77**	53.51	**61.31**	**78.11**	**60.36**	**68.10**	76.27	62.32	68.59
DeepFP(t+fft)	55.38	**58.71**	57.00	71.20	59.69	64.94	**81.32**	**65.07**	**72.29**
DeepFP(t)	65.70	44.12	52.79	67.79	48.63	56.63	80.73	56.39	66.40
PCA	40.90	19.60	26.50	42.20	18.32	25.55	41.53	18.99	26.07
LDA	55.52	12.90	20.93	59.97	12.08	20.11	56.93	11.92	19.71

D. Evaluation Metrics

In our experiments, each segment in the test dataset is evaluated in both *Alien* and 2) *Known user* scenarios. In order to evaluate a segment in the *Alien user* scenario, all other segments pertaining to the same user are removed and the distance between its fingerprint and all those of other segments in the test set are calculated which yields the closeness of the segment to other users. The segment is correctly identified in this scenario if all distances are greater than the pre-specified threshold τ. In other words, the algorithm does not assign that segment to any user. On the other hand, in the *Known user* scenario, the distance between the segment fingerprint and those of the same user must be less than the threshold and minimum among its distances from all other users. Otherwise, it is counted as a miss when it comes to calculating the identification accuracy.

We report the performance in terms of the accuracy of the identification of *Known users* (Positive accuracy) and *Alien users* (Negative accuracy). **Positive accuracy** (P) is the number of segments in the test set which pertain to a user correctly verified to be in the test set divided by the total number of such segments in the test set. **Negative accuracy** (N) is the ratio of correctly verified segments corresponding to a new user. To create a single final measurement for evaluating different techniques, we also report H **accuracy** which is the harmonic mean of P and N to interpret the test's accuracy. H is calculated as $H = 2 \cdot \frac{P \cdot N}{P+N}$.

E. Performance Evaluation

The performance of DeepFP and all baselines are presented in Table I. The results are reported for three different number of segment sizes:$\{10, 20, 40\}$ seconds and the best accuracy for each case is depicted in bold. As it is shown in the table, DeepFP and its variants outperform PCA and LDA. One of the advantages of DeepFP over linear techniques like PCA and LDA is that it leverages a recurrent layer which is capable of detecting both micro and macro patterns.

Among the variants of DeepFP, DeepFP(t+fft) gives the best performance while DeepFP(t) and DeepFP(fft) also show promising accuracy. DeepFP(fft) is better than DeepFP(t) which shows the superiority of the frequency domain compared to the time domain in detecting patterns. The combined time and frequency domain features also provide better performance compared to the time or frequency domain features on their own. While the time domain is inferior to the frequency domain, it still contains some useful information which may have been lost in the frequency domain representation. Moreover, as the size of segments increases, the accuracy gets better, which shows that patterns are more detectable within longer-sized segments.

Over the course of our experiments, we noticed that training DeepFP(t+fft) is approximately three times slower than DeepFP(fft). Consequently, although DeepFP(t+fft) outperforms DeepFP(fft); we use DeepFP(fft) (referred to just as DeepFP) for the following experiments considering that difference in their accuracy is not significant.

F. Model Analysis

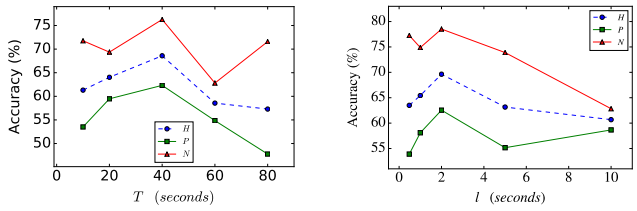
We explore the effectiveness of different variants of neural networks. These variants differ by the type of neural network used in the modeling part of our proposed system. We experimented with LSTM, MLP, RNN and GRU. The number of hidden states for LSTM, RNN, and GRU are all set to 256. The MLP consists of two fully-connected hidden layers of 256 hidden nodes with a ReLU activation function. The experiment is done on two segment sizes of 20 and 40 seconds while all the other parameters are set to the ones given in Section V-C.

We noticed that recurrent-based models with at least 64.01% accuracy give better performance than MLP with an accuracy of at most 41.70%. This likely reflects the capability of recurrent models to detect patterns within sequential data which MLP lacks. Among recurrent-based models LSTM gives the best performance; however, it is slower than others due to its larger number of parameters. We also experimented with both sources of sensors (accelerometer and gyroscope) and came to the result that it helps the performance rather than using only one source. Moreover, accelerometer data source seems to have more informative data to detect behavioral patterns that that of gyroscope due to the model's slightly better corresponding accuracy.

G. Parameter Sensitivity

We analyze the effect of different parameters of DeepFP on its performance which are the segment size and the window

size. The performance of DeepFP for different values of these parameters is illustrated in Figure 2. As the length of segments and windows grows, the accuracy of the model increases until they reach their optimum values and then the performance gradually decreases. One possible reason for this fluctuation is that as the size of segments and windows increases as more information is available within each window and thus more micro patterns can be detected, while the model neglects more macro patterns. It is also possible that as the segments and windows get smaller than a threshold, we miss many micro patterns and thus the accuracy of DeepFP decreases. There exists a trade-off between the amount of detected micro and macro patterns which is achieved by the optimum values of segment and window sizes which are 40 and 2 seconds respectively.



(a) (b)

Fig. 2: The sensitivity of DeepFP for different parameters in terms of accuracy: (a) segment size T, and (b) window size l.

H. Feature Analysis

In Figure 3, we show the effectiveness of DeepFP features compared to time and frequency features in distinguishing different users from each other. We randomly select five users and plot their segments in time, frequency and the DeepFP feature space. They are visualized in a 2-dimentional space using the t-distributed stochastic neighbor embedding feature reduction algorithm (t-SNE) [29]. As shown in Figure 3, fingerprints produced by DeepFP can easily separate most of the segments from different users and thus the feature space learned by our model is more discriminative than that of time and frequency. Additionally, the frequency feature space is slightly better than the time domain but not discriminative enough to be effective at this task.

VI. Conclusion

DeepFP provides a deep model to identify different users via their behavioral patterns while interacting with a mobile phone. DeepFP is employable in real world scenarios where the number of users is too large and re-training the model for new users is challenging. By generating fingerprint vectors from motion sensors which can reliably discriminate between different users, the addition of new users to the system does not necessitate re-training the model. As the fingerprint of a user can get generated on the user's mobile phone, there is no need to send user's data to the central server and it can help to preserve user's privacy. This approach is also suitable for tasks beyond fingerprinting for user tracking, as it can be used for the authentication task, where a DeepFP can be used to increase confidence that a user is (or is not) the intended user of a given device without the need for a substantial amount of training data.

Acknowledgment

We would like to thank the anonymous reviewers for their extensive and helpful feedback. We would also like to show our gratitude to Dr. Jason Polakis who provided comments that greatly improved the manuscript. This material is based upon work supported in part by the National Science Foundation under Grant Nos. CNS-1351058 and CNS-1409868. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The first two authors contributed equally to this work.

References

[1] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies*, vol. 6205. Springer, 2010, pp. 1–18.

[2] P. Laperdrix, W. Rudametkin, and B. Baudry, "Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 878–894.

[3] Apple Computer, "Engineering privacy for your users," https://developer.apple.com/videos/play/wwdc2016/709/, 2016.

[4] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. ACM, 2014, pp. 1054–1067.

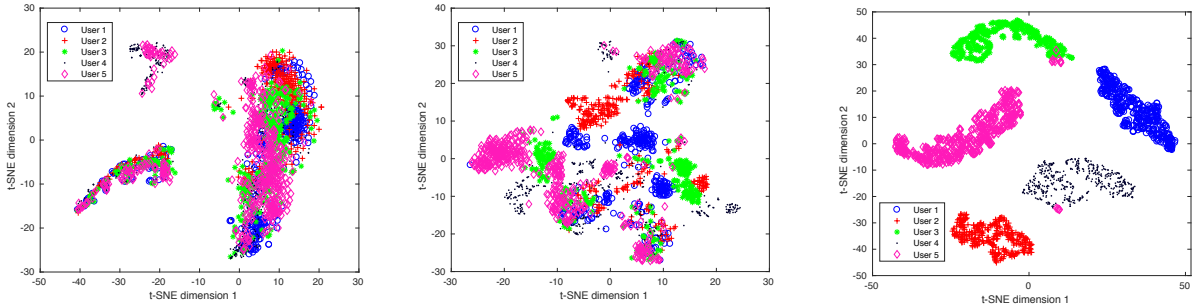
[5] J. Hu, J. Lu, and Y.-P. Tan, "Discriminative deep metric learning for face verification in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1875–1882.

[6] V. Noroozi, L. Zheng, S. Bahaadini, S. Xie, and P. S. Yu, "Seven: deep semi-supervised verification networks," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 2571–2577.

[7] S. Bahaadini, V. Noroozi, N. Rohani, S. Coughlin, M. Zevin, and A. K. Katsaggelos, "Direct: Deep discriminative embedding for clustering of ligo data," in *Proceedings of the 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018.

[8] H-MOG, "H-mog data set - a multimodal data set for evaluating continuous authentication performance in smartphones," http://www.cs.wm.edu/ qyang/hmog.html.

[9] N. Zheng, K. Bai, H. Huang, and H. Wang, "You are how you touch: User verification on smartphones via tapping behaviors," in *Network Protocols (ICNP), 2014*



(a) time subspace (b) frequency subspace (c) DeepFP fingerprint subspace

Fig. 3: t-SNE plots of different feature spaces for 5 randomly selected users.

IEEE 22nd International Conference on. IEEE, 2014, pp. 221–232.

[10] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[11] A. Das, N. Borisov, E. Chou, and M. H. Mughees, "Smartphone fingerprinting via motion sensors: Analyzing feasibility at large-scale and studying real usage patterns," *arXiv preprint arXiv:1605.08763*, 2016.

[12] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[15] W.-H. Lee and R. B. Lee, "Multi-sensor authentication to improve smartphone security," in *Information Systems Security and Privacy (ICISSP), 2015 International Conference on*. IEEE, 2015, pp. 1–11.

[16] C. Bo, L. Zhang, X.-Y. Li, Q. Huang, and Y. Wang, "Silentsense: silent user identification via touch and movement behavioral biometrics," in *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013, pp. 187–190.

[17] C.-C. Lin, C.-C. Chang, D. Liang, and C.-H. Yang, "A new non-intrusive authentication method based on the orientation sensor for smartphone users," in *Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference on*. IEEE, 2012, pp. 245–252.

[18] S. Amini, V. Noroozi, A. Pande, S. Gupte, P. S. Yu, and C. Kanich, "Deepauth: A framework for continuous user re-authentication in mobile apps," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018, pp. 2027–2035.

[19] M. Gadaleta and M. Rossi, "Idnet: Smartphone-based gait recognition with convolutional neural networks," *Pattern Recognition*, vol. 74, pp. 25–37, 2018.

[20] H. M. Thang, V. Q. Viet, N. D. Thuc, and D. Choi, "Gait identification using accelerometer on mobile phone," in *Control, Automation and Information Sciences (ICCAIS), 2012 International Conference on*. IEEE, 2012, pp. 344–348.

[21] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "Deepsense: A unified deep learning framework for time-series mobile sensing data processing," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 351–360.

[22] K. Shah, S. Kopru, and J. D. Ruvini, "Neural network based extreme classification and similarity models for product matching," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, vol. 3, 2018, pp. 8–15.

[23] B. Boashash, *Time-frequency signal analysis and processing: a comprehensive reference*. Academic Press, 2015.

[24] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning*, 2015, pp. 2342–2350.

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] Z. Sitová, J. Šeděnka, Q. Yang, G. Peng, G. Zhou, P. Gasti, and K. S. Balagani, "Hmog: New behavioral biometric features for continuous authentication of smartphone users," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 877–892, 2016.

[27] C. R. Rao, "The use and interpretation of principal component analysis in applied research," *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 329–358, 1964.

[28] K. Fukunaga, *Introduction to statistical pattern recognition*. Academic press, 2013.

[29] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.