Preserving Privacy in Personalized Models for Distributed Mobile Services

Akanksha Atrey
University of Massachusetts Amherst
aatrey@cs.umass.edu

Prashant Shenoy
University of Massachusetts Amherst
shenoy@cs.umass.edu

David Jensen
University of Massachusetts Amherst
jensen@cs.umass.edu

***Abstract*—The ubiquity of mobile devices has led to the proliferation of mobile services that provide personalized and context-aware content to their users. Modern mobile services are distributed between end-devices, such as smartphones, and remote servers that reside in the cloud. Such services thrive on their ability to predict future contexts to pre-fetch content or make context-specific recommendations. An increasingly common method to predict future contexts, such as location, is via machine learning (ML) models. Recent work in context prediction has focused on ML model personalization where a personalized model is learned for each individual user in order to tailor predictions or recommendations to a user's mobile behavior. While the use of personalized models increases efficacy of the mobile service, we argue that it increases privacy risk since a personalized model encodes contextual behavior unique to each user. To demonstrate these privacy risks, we present several attribute inference-based privacy attacks and show that such attacks can leak privacy with up to 78% efficacy for top-3 predictions. We present *Pelican*, a privacy-preserving personalization system for context-aware mobile services that leverages both device and cloud resources to personalize ML models while minimizing the risk of privacy leakage for users. We evaluate *Pelican* using real world traces for location-aware mobile services and show that *Pelican* can substantially reduce privacy leakage by up to 75%.**

***Index Terms*—cloud-based mobile services, personalized ML models, privacy, deep learning, context-awareness**

I. Introduction

The ubiquitous nature of smartphones and smart devices, such as wearables, have led to a plethora of online mobile services in various domains including fitness, entertainment, news and smart homes. Such mobile services tend to be distributed between the end-device and the cloud with front-end components running on the devices as mobile applications and back-end components running on cloud servers. Modern mobile services are often context-aware to provide tailored content or service to users based on their current context. For example, it is common for a restaurant recommendation service to use location as its context when recommending nearby eateries. While the use of *current* context in mobile services is common, mobile services have begun to use machine learning (ML) models to predict *future* contexts (e.g., a user's next or future location(s)) and provide tailored recommendation based on these prediction (e.g., suggest directions or store closing time of predicted future location).

Machine learning has been used in mobile services for tasks such as next location prediction [1], medical disease detection [2] and language modeling [3]. The popularity of deep learning has established the use of aggregated data from a large number of users to train and deploy a general ML model that makes predictions for context-aware services for a broad range of users. A more recent trend in the field is to use personalized models on a per-user basis rather than a general model to further improve the efficacy of the service. In this scenario, rather than using a single ML model for all users, a model is personalized for each user using training data specific to the user. For instance, a user's frequently visited locations in a mobile service or a user's viewing history in a streaming service can be used to develop personalized ML models.

While model personalization is a growing trend in mobile and Internet of Things services, in this paper, we examine the implications of such an approach on the privacy of individuals. We argue that personalized ML models encode sensitive information in the single-user context traces used as training data and mobile services that use such personalized models can leak privacy information through a class of privacy attacks known as model inversion. Model inversion attacks exploit a trained ML model to infer sensitive attributes [4]. While ML researchers have studied inversion attacks in other contexts, they have not been studied or demonstrated for time-series models that are commonplace in mobile applications. Our work formalizes and demonstrates such attacks for personalized mobile services by showing how they can leak sensitive context (i.e. location) information about a user. To the best of our knowledge, privacy implications of personalized models in distributed mobile services have not been previously studied.

Motivated by the need to ensure the privacy of personalized ML models, we present *Pelican*, an end-to-end system for training and deploying personalized ML models for context-aware mobile services. Our system enhances user privacy by performing sensitive personalized training on a user's device and adding privacy enhancements to personalized models to further reduce and prevent inversion attacks from leaking sensitive user information. Our system is also designed to allow low overhead model updates to improve model accuracy while safeguarding privacy. Finally, our system leverages the device and cloud architecture of mobile services when personalizing ML models to enhance user privacy. In design and implementation of *Pelican*, we make the following contributions:

C1 We adapt low-resource transfer learning methods to train and execute personalized ML models on resource-

constrained mobile devices for mobility applications. Our approach utilizes the inductive biases of a multi-user ML model and tailors it to a distinct user using their limited context traces. Our work draws inspiration from existing work on transfer learning-based personalization of language models [3].

C2 We formalize practical inference-based privacy attacks on personalized models using model inversion [4]. We consider ways in which an adversary can reconstruct private historical information using only trained personalized mobility prediction models. Our work formalizes model inversion attacks for time-series based ML models with application in the mobility domain.

C3 We quantify the efficacy of these privacy attacks on mobile services that use personalized models. Our findings demonstrate that such attacks can leak private historical mobility patterns with up to 78% accuracy for top-3 predictions. We find that the leakage is higher for smaller spatial scales and independent of user mobility behavior.

C4 We present the design of *Pelican*, an end-to-end privacy preserving personalization framework. We propose a robust enhancement to mitigate inference-based privacy attacks based on scaling the output probability distribution at inference time. We empirically evaluate *Pelican* on low-level and high-level spatial mobility scales using a campus dataset and show that *Pelican* is able to reduce privacy leakage up to 75%.

II. Background

In this section, we present background on context-aware mobile services and the use of ML models in such services.

Context-Aware Mobile Services. Our work considers mobile services whose service components are distributed across mobile devices and a back-end cloud. It is typical for mobile services to be context-aware and tailor the service based on current or future contexts. In recent years, context-aware mobility applications, such as location-based social networking and ride-sharing applications, have gained popularity. Context can be defined as any information used to characterize interactions with the environment or situation of an entity and can be broadly categorized into temporal, spatial and social. A common type of context-aware service utilizes the user's current or future location to offer location-aware mobile services. Unless specified otherwise, our work assumes location to be the primary context used by the distributed mobile service.

Mobility Prediction. In addition to using current context such as location, many services now use next location prediction techniques to predict future location(s) that a user will visit and offer recommendations based on future contexts. For instance, a mapping service may predict commute times to the next location a user is predicted to visit. Next location prediction techniques capture the spatial and temporal correlations between human mobility patterns. Since humans tend to follow particular routines and habits, learning their mobility behaviors can assist many domains from recommendation systems to urban design. Human mobility can be defined through a series of location and time-varying features. Consider a set of features $x_t = \{l, e, d\}$ with location l, entry time e and duration d at time t. The mobility prediction problem can be defined as follows: given a set of previous sequences $s_u = \{x_1, x_2, ...x_t\}$ for user u, estimate location l_{t+1} of user u at the next time step.

Time-Series ML for Next Location Prediction. Prior work in next location prediction has focused on using variants of Markov models, Hidden-Markov models and tree-based classification models to learn the sequential nature of mobility [5], [6]. With the emerging capabilities in deep learning to handle temporal or spatial input, recurrent neural networks (RNN) have been proposed for mobility prediction [7]. RNNs have the ability to capture sequential data where each sample is dependent on its previous samples. More recently, a variant of RNNs, long short term memory (LSTM) models [8] have shown state-of-the-art performance in predicting human mobility [1], [9]–[11]. Unlike RNNs, LSTMs have the ability to learn and remember long-term dependencies in the data. Deep learning-based models generally employ mobility trajectories of many users to learn generic human mobility patterns and are capable of handling large prediction spaces typical of general mobility models.

Model Personalization. A common approach for using ML models in mobile services (e.g., for predicting future contexts) is to train a general ML model using aggregated training data from a larger number of users. Such a model encodes behavior of a large group of users and can predict the future behavior of a user who resembles one in the training set. A recent trend, however, is to employ a personalized model that is designed for a specific user over the use of a general model. Personalized models can encode specific behavior exhibited by an individual user and offer better efficacy over an aggregated model. In recent years, machine learning methods for personalization have been proposed in various domains including autonomous vehicles [12], health [13], and natural language processing [3]. Recently, Sarker et al. explored the effectiveness of ML models for predicting personalized context-aware smartphone usage [14]. They evaluate numerous ML algorithms and find that tree-based models, such as random forests, are the most effective for building personalized context-aware models. Personalized modeling in mobility has been generally conducted via Markov models [5]. More recently, Feng et al. developed personal adaptors for personalized modeling with LSTMs [11].

Machine Learning Privacy. Machine learning models are vulnerable to privacy attacks and our work argues that model personalization increases privacy risks for users. Two of the primary privacy attacks in ML are *membership inference attacks* [15] and *attribute inference attacks* [4]. Membership inference attacks aim at inferring whether a data sample was present in the training set. Given a model M and some data point x, the goal is to infer whether M used x during training. This attack is particularly problematic when using sensitive data sets. For instance, if a ML model is trained on a cancer data set and an adversary is able to infer whether a user

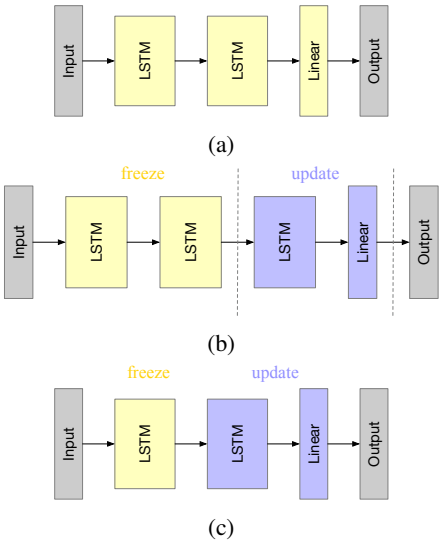


Fig. 1: Next location prediction architectures: (a) traditional general model; (b) transfer learning-based feature extraction model; and (c) transfer learning-based fine tuned model.

was in this data, it will reveal the user's health status. In this work, we focus on attribute inference attacks, namely a model inversion attack. Model inversion attacks aim at inferring sensitive features using a trained model. Given a model M and some features $\{x_2, x_3..., x_n\}$, the goal is to reconstruct the value of some sensitive feature x_1. This is problematic when the data set contains sensitive features such as the location of a user. Model inversion attacks were first proposed by Fredrikson et al. [16] to exploit linear regression pharmacogenetic models to infer patient genotype. There have been various subsequent papers on understanding the broader risk of such attacks [4], [17]–[21]. Wu et al. proposed a game-based formalization of inversion attacks for any ML model yet claimed that privacy leakage from such attacks is context dependent [17]. Our work formalizes model inversion attacks for time-series applications with a focus on mobility. We focus on reconstructing users' historical mobility patterns using a next location prediction model.

III. Personalized Models and Privacy Implications

In this section, we first describe our approach for model personalization based on transfer learning of deep learning models and then describe our model inversion privacy attack on such personalized models.

A. ML-driven Next Location Prediction

Predicting the next location(s) based on historical locations is a fundamental mobility task that is useful in a broad range of mobile services. We describe three approaches based on deep learning to perform next location prediction.

1) LSTM-based General Model: The traditional approach has been to use historical trajectories, temporally extended sequences of locations, of many different users to train a deep neural network that predicts the next location of any given user. Early approaches were based on RNNs while the state-of-the-art approaches use LSTMs [8] to capture both the short-term and long-term dependencies in user mobility patterns. Figure 1a illustrates an example architecture of a LSTM model with two LSTM layers followed by a linear layer. Since training deep models, including LSTMs, requires a large amount of training data, a common approach is to use historical trajectories of many users to train an accurate but general model that performs next location prediction [1], [7], [9], [10], [22].

2) Personalized Models: While a general LSTM model can learn correlations in mobile behavior across users and perform well across a range of users that behave similarly, they are less effective for individual users who exhibit idiosyncratic or dissimilar behavior. To address this issue, researchers have proposed to train personalized models for users to capture their unique behavior [11]. While a single model is used for all users in case of a general model, personalization requires that a unique model is learned and deployed for each user.

A LSTM model similar to a general model can be used for training personalized models. In this case, historical trajectories *from a single user* are used to train each model. The advantage of model personalization is that it can yield more accurate user-specific predictions. However, deep learning models require a large amount of single user data to train each personalized model (unlike a general model where less single user data suffices due to the availability of training data from many similar users).

3) Transfer Learning-Based Personalization: In our work, we assume a different approach for model personalization that overcomes some of the limitations of the above methods. Our approach involves first training a general model for next location prediction using training data from multiple users. Then it uses transfer learning to personalize the general model for a new user using their historical data. The advantage of personalizing an already trained general LSTM model using transfer learning is that it requires less single user historical data than training one from scratch.

The goal of transfer learning is to transfer knowledge learned from solving one task to assist another [23]. Existing areas that employ transfer learning, such as computer vision and natural language processing, typically have a fixed domain size between source and target tasks. However, the domain of the multi-user model can differ from the domain of the single user data for next location prediction. For instance, a general mobility prediction model that is trained for New York City will have a different domain from a user who lives in Boston. In this work, we assume that the target single-user domain is a subset of the source multi-user domain. Assume the source domain is D_s and target domain is D_t, where $D_t \subseteq D_s$. Prior to applying transfer learning, we transform the target data by extending the domain with $D_s - D_t$. In our case, this implies introducing new categories (e.g., $D_s - D_t$) to the existing one-hot encoded location categories in the target data. This simplifies the transfer learning process by equalizing the source and target domains. Employing heterogeneous transfer

learning methods for mobility is a direction for future work.

There are two popular methods for transfer learning, either of which can be used to personalize a general model using a small amount of user data.

Feature Extraction. One popular method to conduct transfer learning is to employ the general model as a feature extractor for learning the generic patterns relevant to the task. The layered architecture of deep learning models assist in learning different features or correlations within the data at different layers. Since the general model takes as input the trajectories of many users, it learns a representation of the generic mobility pattern of the users. The intuition behind feature extraction is to exploit and build on top of the representation learned by the generic model. This is conducted by using the primary representation layers of the trained general model (e.g., first two LSTM layers in Figure 1a) and adding a surplus layer or a new shallow model before the final linear layer to learn specific patterns from the single user data. This method requires re-training the model with single-user data, but only updating the parameters of the newly appended shallow model. To ensure that only the newly appended shallow model is updated and the generic patterns are not lost during the training process, the weights of the general model layers prior to the shallow model are frozen. In our work, we stack another LSTM layer before the output layer to capture the patterns unique to the user as shown in Figure 1b.

Fine Tuning. Another popular transfer learning approach considers fine tuning the trained general model instead of building on top of it. The initial layers in a deep learning model often focus on generic patterns and the latter layers focus on specific patterns relevant to the task at hand. During transfer learning, the goal typically is to transfer the generic features and learn the specific patterns based on the target data (e.g., single-user trajectory). To do so, one method is to freeze the initial layers and re-train the latter layers with single user data. Figure 1c shows an example of such a model. The particular number of layers to re-train or fine-tune depends on the nature of the data. With plenty data, more layers can be re-trained whereas with sparse data, often the case with single-user trajectories, minimizing this number can be better due to the risk of overfitting. In our work, we re-train and update parameters of the second LSTM layer and linear layer using single user data.

B. Privacy Attacks on Personalized Models

As noted in Section II, ML models are vulnerable to privacy attacks. A particular type of ML privacy attack is a model inversion attack that exploits a trained ML model to infer values of sensitive attributes in the training data [4]. While inversion attacks have been studied in other contexts, prior work has not explored inversion attacks on time-series based ML models, and specifically, context-aware services that use time-series trajectories of contexts such as location history.

Intuitively, a model inversion attack takes a trained model as a black box and a possible output y produced by the model to learn one or more model input features that produce this output. A simple model inversion attack exploits confidence values and prior knowledge of the sensitive variable by picking the value that maximizes the model's confidence of y weighted by the prior [4]. In case of a next location prediction model, it implies taking a predicted next location (model output) to learn one of the previous locations (model input) visited by the user. This is concerning given the sensitivity of location data (e.g., visit to a hospital can leak privacy). The goal is to not reveal more than needed for the service to operate.

Model inversion attacks have greater privacy implications for personalized models than general models. Since a general model is trained using data from many users, leaking a previously visited location as present in the training data may not directly reveal private information of a specific user. However, an inversion attack on a personalized model directly reveals prior locations visited by a specific user, which can leak sensitive information about that user. In the rest of this section, we formalize and describe a model attack inversion attack on personalized time-series next-location models.

1) Threat Model: We consider a system which consists of a location-aware mobile application that collects sequences of data $x_t = \{f_1, f_2, ..., f_k\}$ with k features at each time step t. This system consists of the following entities:

Contributors. We assume there exists a set $\mathcal{G}$ of unique users who allow their data to be used to train a multi-user ML model, M_G, for next location prediction. These users serve as *contributors* for M_G.

Users. We consider a set of honest unique users $\mathcal{P}$, disjoint from $\mathcal{G}$, that use the location-aware application. We assume all users in $\mathcal{P}$ employ a transfer learning-based personalization method (see Section III-A3) and general model M_G to build personal models. These users protect their data by keeping it local and only allowing the service provider black-box access to the personal ML model.

Service Provider. We consider a service provider $\mathbb{S}$ that hosts the location-aware mobile application. $\mathbb{S}$ has access to the data sequences of users in set $\mathcal{G}$ using which it trains M_G, and only black-box access to trained personal models of users in $\mathcal{P}$. We assume $\mathbb{S}$ has the ability to query and observe the model output and associated confidence scores for all classes. We consider $\mathbb{S}$ to be a honest-but-curious adversary that attempts to learn historical mobility pattern of users in $\mathcal{P}$ using their personal ML models.

Since our focus is on privacy rather than security, we do not consider security threats from external adversaries who may break into the system and steal private data or models.

2) Proposed Privacy Attack: Our focus in this paper is on attribute inference attacks using model inversion. The proposed model inversion attack follows the basic premise as described earlier. We assume that all personal models output confidence scores (probabilities) for all classes. This is a typical assumption in mobility applications, particularly when the focus is on getting the top k most likely next locations rather than a single next location. Let $p = (p_1, ..., p_m)$ be the marginal probabilities of the sensitive variable that can take m values. For instance, if the sensitive variable is building-level

TABLE I: Descriptions of different adversaries with the components they have access to and their goal. M_p refers to a user's personalized model, p refers to prior knowledge, x_{t-1} and x_{t-2} are inputs to M_p, and l_t is the output of M_p.

Adversary	**Adversarial Knowledge**					**Goal**
	M_P	p	x_{t-1}	x_{t-2}	l_t	
A1	✓	✓	-	✓	✓	l_{t-1}
A2	✓	✓	✓	-	✓	l_{t-2}
A3	✓	✓	-	-	✓	l_{t-1} or l_{t-2}

location, the marginal probability p_i will reflect how often building i is visited. The novelty in our work arises from the formalization of this attack from a time-series context.

We assume that adversarial access to features is limited by time. That is, an adversary has access to all or no features within a sequence for a given time step. For simplicity, we further assume that there is a single sensitive variable at each time step (e.g., location l) for all adversaries. Table I presents descriptions of different adversaries with the features they have access to and their goal. We assume all adversaries have access to some location of the user. A honest-but-curious service provider can simply observe the output of the personal models (i.e., l_t) or gather such information from other context-aware applications, mobile cookies, third-party applications or location-based social networks. A1 and A2 represent the simplest adversaries which have access to all features except features at time x_{t-1} or x_{t-2} with the goal of correctly identifying l_{t-1} and l_{t-2} respectively. Note, these adversaries require some historical external information namely all features at time t–2 and t–1 respectively. Adversary A3 represents an adversary who has limited access to historical sequences but has information on model output or some location l_t.

A popular form of model inversion attacks require enumeration over values of the sensitive variable(s). The simplest and most computationally expensive form of enumeration for time-series data is a *brute force* method where an adversary enumerates through all the features in an unknown sequence x_t. Since deep learning models learn a differentiable mapping between the input and the output, it is also possible to reconstruct the input using the output through backpropagation and *gradient descent*. Backpropagation is used in deep learning to calculate the gradient of the loss function with respect to the parameters of the model and gradient descent allows a descent or step in the direction that optimizes the loss function through the gradient. We employ this algorithm to reconstruct the input, sequences x_{t-2} and x_{t-1}, by iteratively transforming a candidate input towards the values that maximize the correct output. To deal with the large output space typical in mobility domains, we also add the notion of temperature scaling. Temperature, $\mathcal{T}$, is a hyperparameter that controls the variability in prediction space by scaling the raw probabilities (i.e., logits) before applying softmax. The logits (z_i) are divided by this term before applying the softmax function:

$$p_i = \frac{\exp\left(z_i/\mathcal{T}\right)}{\sum_i \exp\left(z_i/\mathcal{T}\right)} \tag{1}$$

We use this as a method to soften the candidate input variables during gradient descent such that they are one-hot encoded and represent discretized values.

Additionally, we propose an enumeration method that employs the *time-based dependence* between the features. Considering that mobile devices are consistently with users, we can assume that there exists cross-correlation between consequent sequences and continuity (e.g., no gaps in time periods). Thus, we can use smart enumeration techniques that take advantage of these correlations by enumerating through only certain features and using cross-correlation to infer the rest. This method is dependent on the nature of the input features and works for numerical time-varying features. For example, if we assume a sequence consists of location (l), duration at location (d), and entry time at location (e), for adversary A1, we can enumerate through d_{t-2} and l_{t-2} and compute e_{t-2} from knowledge of e_{t-1} and d_{t-2} (e.g., $e_{t-2} = e_{t-1} - d_{t-2}$). Moreover, to minimize the search space, we propose identifying the user's locations of interest. Since the adversary is assumed to have black-box access to the model, we propose observing the output for a few instances and selecting only locations with confidence greater than or equal to some threshold (i.e. 1%). This will minimize the search space substantially, particularly since the personalized model includes all locations in a given proximity, instead of only those captured in the user's data due to the domain equalization mentioned in Section III-A3.

IV. Privacy Leakage from Inversion Attacks

In this section, we empirically evaluate the efficacy of the model inversion privacy attack presented in Section III-B.

A. Experimental Setup

Data. We employ a campus-scale WiFi dataset from September to November 2019. This data consists of 156 buildings that are connected by 5104 HP Aruba access points (APs). Each AP event includes a timestamp, event type, MAC address of the device and the AP. Since the WiFi network requires all users to authenticate themselves, each event can be associated with a user. For this work, all user information is anonymized using a hashing algorithm.

Using well known methods for extracting device trajectories from WiFi logs (e.g., [10]), we extract fine-grained mobility trajectory of 300 users spanning over 150 buildings and 2956 APs. We filter the data to consist of only on-campus students by assessing whether users stay in a dorm on a typical weekday night. The final processed data set includes sequences of four features for each user: *session-entry* (e), *session-duration* (d), *building* (l), and *day-of-week* (w). Note, *session-entry* is discretized into 30 minutes intervals and *session-duration* is discretized into 10 minutes intervals to reduce the variability. Duration is also capped at 4 hours since less than 10% of users spend more time in a single building [10].

Task. We focus on next-location prediction using historical trajectories. Let $x_t = [e_t, d_t, l_t, w_t]$ be a sequence at time t. Then, let the ML model be $M : x_{t-2}, x_{t-1} \rightarrow l_t$. That is, the ML model takes as input two sequences and outputs the next

TABLE II: Runtime of attack methods for 100 users.

Method	Runtime (hours)
Brute Force	82.18
Gradient Descent	6.27
Time-Based	0.68

location. We employ both building-level and AP-level spatial scales for our experiments. Location l is considered to be a sensitive variable.

Models. We employ trajectories of 200 users to train the general LSTM as described in Section III-A1. 80% of the data is used for training and 20% is used for testing. We perform grid search on time-series based 5-fold cross validation to select the optimal hyperparameters for the model. The general LSTM is trained using a learning rate of $1e{-}4$ with a weight decay of $1e{-}6$ and hidden layer size of 128. We use batches of size 128 with a dropout rate of 0.1 between the LSTM layers. To learn personalized models, without loss of generality, we employ transfer learning-based feature extraction (TL FE) (see Section III-A3). We train individual personalized models for 100 unique and distinct users. We perform grid search using 3-fold time-series cross validation for hyperparameter selection.

Measures. We employ *top-k accuracy* as an evaluation metric. The goal is to identify the top-k most likely locations from the model output and assess whether the true location is a subset of that.

B. Analysis of Privacy Attack

We analyze the proposed privacy attack on 100 distinct users. We use time-based dependence and adversary A1 as our default attack method and adversary respectively, and perform all experiments on building spatial level unless otherwise stated. For all experiments, attack accuracy is defined as the percentage of historical locations correctly identified.

1) Impact of attack type: We compare the two proposed attack methods, time-based enumeration and gradient descent with temperature scaling, with brute force. Figure 2a contains an evaluation of the attack methods discussed in Section III-B. As expected, the brute force method performs well, reaching 79.64% attack accuracy for top-3 predictions. Our proposed time-based method performs equivalently to the brute force method with attack accuracy growing as k increases. However, the gradient descent method is the least effective at constructing historical mobility patterns with attack accuracy of less than 16%. We hypothesize this is due to the large domain size and discrete nature, instead of continuous, of mobility locations which results in an inaccurate reconstruction of the historical data.

Despite the similar performance, the brute force and time-based enumeration methods differ substantially in computational complexity. The runtime of the brute force method is over 120 times that of the time-based method suggesting that the time-based attack is highly efficient to launch in practical settings. Table II contains runtimes of the three methods.

2) Impact of adversarial knowledge: The results shown in Figure 2b illustrate the impact of adversarial knowledge from Table I on the attack. Despite the differing levels of adversarial knowledge, all adversaries perform effectively and equivalently at reconstructing historical mobility patterns. Interestingly, adversary A3's attack capabilities do not degrade despite the lack of adversarial knowledge. This illustrates that even with limited prior information on historical time steps, an adversary can effectively perform a model inversion attack.

3) Impact of prior information: All experiments thus far assume that the adversary has access to the true marginal probabilities of the sensitive variable. However, this is unlikely to be known by a typical adversary. In reality, an adversary can get access to the most probable value(s) of the sensitive variable but not know exact probabilities. We attempt to *estimate* the marginal probabilities p in this manner by assigning a high probability (e.g., 75%) to the most probable value and equally distributing the remaining probability among the other values. The adversary can also easily observe the output of the target model for a period of time and *predict* p. Figure 2c demonstrates the impact of different p generation methods, namely *true*, *none*, *predict* and *estimate*.

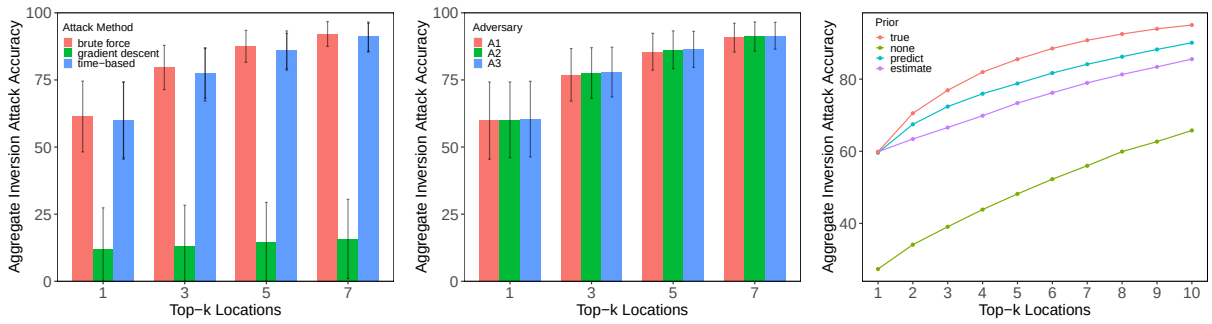
The results in Figure 2c confirm the importance of using p during the attack; without p, the attack is less effective. However, the attack is not sensitive to the precision of p. The *true* method results in the highest attack effectiveness across k whereas predicting or estimating p results in a 5-10% degradation in attack efficacy. The difference between true, predict and estimate methods grows as k increases. Naturally, among these three, the effectiveness of the estimate method grows the slowest as k increases, due to its highly skewed probability estimates.

4) Impact of mobility spatial levels: Mobility spatial levels (the spatial resolution) can differ based on the task definition. Thus far, all experiments were evaluated at a building-level scale. To understand the impact of a fine-grained spatial scale, we run the attack at the scale of access points (APs). There are 2956 APs in our data set.

The results in Figure 3a show that the attack leaks less privacy at the AP scale when compared to building scale. We hypothesize this is due to the large domain size of AP-level models, which makes it difficult to reconstruct historical patterns. Similar to building scale, there is more privacy leakage as k grows. In future work, we would like to consider ways to handle larger spatial scales.

5) Impact of degree of mobility: We also evaluate how characteristics of mobility affect privacy leakage. The degree of mobility varies for different users. Highly mobile users visit many locations and less mobile users tend to visit fewer locations during a given time period. For instance, socially active users may physically move around more than their counterparts. We evaluate how degree of mobility affects attack accuracy in Figure 3b.

The degree of mobility has a weak effect on privacy leakage. Since users tend to spend a majority of their time at a single location [10], it is likely that the attack is less affected by the degree of mobility at less visited locations. These results are supported by a regression analysis; the correlation coeffi-



(a) (b) (c)

Fig. 2: Results of evaluating the efficacy of the privacy attack under varying system configurations: (a) impact of varying attack methods; (b) impact of varying adversarial knowledge; and (c) impact of nature of prior knowledge p.

cients are weak, 0.337 and 0.107 for building and AP level respectively, with statistically significant p-values ($p \leq 0.05$).

6) Impact of mobility predictability: We further evaluate the impact of mobility predictability on attack accuracy. Highly predictable users have highly correlated mobility patterns across time and space. We employ the personalized model accuracy as a proxy for mobility predictability. That is, higher model accuracy implies higher predictability of mobility since the model is expected to capture the correlations in the mobility pattern of the user.

We show results in Figure 3c. Mobility predictability strongly affects privacy leakage for building spatial level. This is not surprising since the attack is based on inverting the model itself; more accurate models more precisely capture mobility patterns which can then be exploited by the attack. These results are supported by numerical results from regression analysis. There is a strong correlation coefficient of 0.804 with a statistically significant p-value ($p = 2.92\text{e}{-2}$). However, we note that the relationship is weak for AP spatial level with a correlation coefficient of 0.078 and insignificant p-value of 0.031. We hypothesize that the distribution of time spent in different APs can explain the variance in attack accuracies for similar target model accuracies seen in Figure 3c.

Key Takeaways: The proposed time-based model inversion attack is computationally efficient and effective at revealing historical mobility patterns with 77.61% accuracy for top-3 estimates even with limited adversarial knowledge and low precision prior p. While the privacy leakage is independent of the mobility behavior of the user, there is a trade-off between model efficacy (i.e. correlation in data) and privacy. Furthermore, models of coarse-grained spatial scales leak more privacy. These results demonstrate that context-aware personalized models can be easily exploited with limited information for users with highly correlated mobility patterns.

V. Privacy Preserving ML Framework for Mobile Services

In this section, we present *Pelican*, a privacy preserving framework for machine learning-based mobile services.

A. System Design

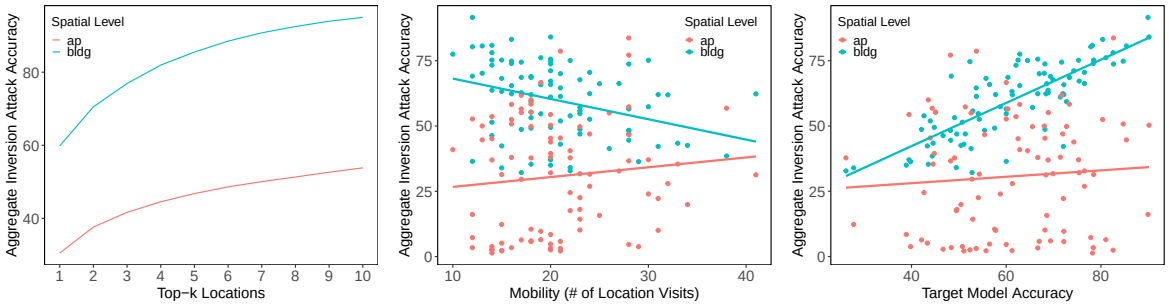
Pelican is a distributed framework for training and deploying personalized ML models for mobile services in a privacy preserving manner. *Pelican*'s architecture is designed to safeguard private training data of individual users, such as historical location trajectories, while learning a personalized model. *Pelican* also incorporates privacy preserving enhancements into the deep learning model itself to thwart model inversion attacks. The framework leverages the device and cloud tiers of distributed mobile services to achieve its goals. Figure 4 depicts the design of *Pelican*.

Pelican comprises of the following key components:

1) Cloud-based Initial Training: The first step in designing a privacy-preserving ML model for mobility is to train a general model, M_G, using training data from multiple users. Since initial training of the model is compute intensive, this component of our framework runs on cloud servers and leverages specialized resources such as GPUs when available. The initial training components invokes a deep learning library on a cluster of cloud servers to train a general model. For example, in case of next location prediction, we train a LSTM-based deep learning model using time-series trajectories of locations visited by 200 users over a duration of two months.

2) Device-based Personalization: Once a general ML model has been trained in the cloud, the next phase personalizes this model for each user using transfer learning. The personalization involves using a small amount of training data for each new user to learn a distinct personalized model, M_P. Since the personal training data contains sensitive private information (e.g., location visits), the training for personalization is executed on the local device rather than the cloud. Retaining all private data on local user-owned devices enhances privacy.

To do so, the general model is downloaded from the cloud to the device and transfer learning is performed on the device using personal training data (e.g., location history of the user). Transfer learning-based personalization can be conducted via feature extraction or fine tuning (see Section III-A3) depending on the nature of the data. If the personal training data is sparse, feature extraction should be used to avoid overfitting. Note that unlike training the general model which is compute



(a) (b) (c)

Fig. 3: Results for two spatial scales with 150 unique buildings and 2956 unique APs: (a) impact of the privacy attack on different spatial scales; (b) impact of degree of mobility on privacy; and (c) impact of mobility predictability on privacy.

intensive and is performed in the cloud, transfer learning is much less compute intensive and can be performed on devices that are resource constrained [3]. This phase also involves adding privacy preserving enhancements to the LSTM model (as discussed in Section V-B).

3) Model Deployment: Once the model has been personalized using transfer learning, it is ready for deployment in the mobile service. Since mobile services can vary in their characteristics, the model can be deployed in two ways.

The first approach is local on-device deployment where the model executes on the device for making predictions. This approach is suitable for mobile services that run largely on devices (e.g., smartphone mobile apps with a lightweight cloud component). Local deployment avoids network latency to the cloud for AI inference and ensures that the model stays on the user's device minimizing the amount of information known by the service provider and consequently enhancing privacy.

The second approach is to deploy the personalized model in the cloud. This approach is suitable for cloud-based services and enables the service to invoke the model in the cloud to provide context-aware service to the user. In this case, even though the model runs in the cloud, its privacy enhancements prevent model inversion attacks (see Section V-B).

4) Model Updates: It is common for production services to periodically re-train the ML model to update it with new training data as it becomes available. In our case, as new personal data becomes available, the transfer learning process can be re-invoked to update the parameters of the personalized model, after which it is redeployed for use by the service.

The framework also allows the general model to be updated in the cloud periodically, but this requires re-running the transfer learning process on the device to re-personalize the model for each user. Due to the higher overheads of doing so, updates to the general model are done infrequently while updates to the personalized model can be done frequently.

B. Privacy Enhancements to Personalized Models

We now present our privacy enhancement to the LSTM model during model personalization that is designed to thwart inversion attacks. Our goal is to protect training data privacy such that adversaries cannot reverse-engineer a black-box personalized model to learn historical mobility patterns.

The proposed attack thrives on the adversary's ability to access the model's output and confidence scores. The enhancement aims to satisfy the following requirements:

1) The personalized model can be accessed by the service provider in a black-box manner. This allows the service provider to query the model.
2) The service provider can access model outputs to get context-aware predictions. The service provider can also access confidence scores to compute the top-k locations.
3) The service provider cannot determine historical mobility patterns by reverse engineering the model.

The proposed enhancement is based on modification of the confidence scores such that the attack space reduces tremendously. Our approach introduces a new layer into the LSTM model between the linear layer and softmax layer that changes the distribution of the confidence scores without compromising model accuracy. This layer takes as input the raw probabilities from the linear layer. Before applying the softmax function to normalize these raw probabilities, this layer scales the probabilities by dividing them with a value $\mathcal{T}$. Note, this is similar to using temperature scaling, a single parameter extension of Platt scaling [24], in deep learning. Temperature is a hyperparameter often used to control the randomness in the predictions (see Equation 1).

In our work, we use the notion of temperature as a privacy tuner to change the sensitivity to the different outputs at inference time only. As the temperature tends to 0, the confidence of the sample with the highest probability tends to 1. Intuitively, this makes the attack more difficult because the confidence scores will be highly insensitive (i.e., close to 0 or 1). With sharper confidence values, the attack space will reduce and adversaries will not be able to reconstruct historical mobility patterns meaningfully. Note, since the order of the confidence values do not change during scaling, the model's accuracy will remain unaffected as long as appropriate precision is used in storing the confidence values.

The enhancement is designed as a user-centric mechanism; we use this parameter as a value that can be determined by the

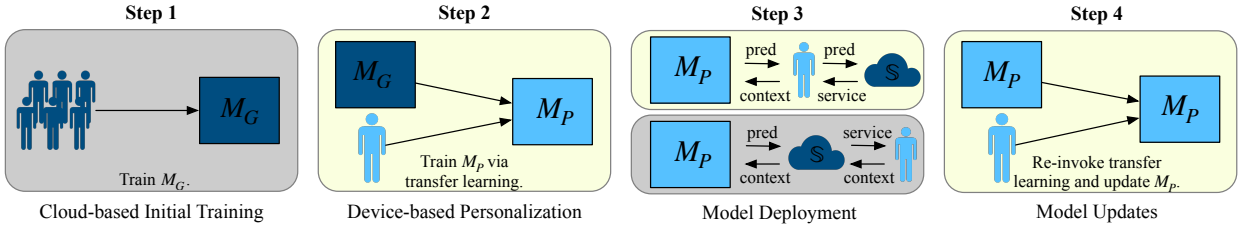


Fig. 4: Overview of the proposed system, *Pelican*. Grey and yellow represents phases that are executed on the cloud and device respectively. Phase 3 can occur in either the cloud or the device depending on the service characteristics. M_G is the general multi-user model, M_P is the personalized model and $\mathbb{S}$ is the service provider.

user. The user can pick a small or large value depending on how much privacy (i.e., insensitivity to the confidence scores) they prefer. We assume the value of the privacy tuner is kept private from the service provider.

C. System Evaluation

1) Prototype and Experimental Setup: To evaluate *Pelican*, we employ the same campus-scale WiFi dataset, next-location prediction task and top-k measure as described in Section IV-A. The system prototype is implemented in Python and all deep learning models are built using PyTorch. The general model in the cloud-based initial training follows the architecture presented in Figure 1a and is implemented using PyTorch's `nn.LSTM`, `nn.Dropout` and `nn.Linear` layers. The transfer learning-based feature extraction method is implemented using PyTorch's `nn.Sequential` container, whereas the transfer learning-based fine tune method involved re-training part of the general model. We assume that the model is deployed in a location-aware application. For model updates, we initialize the model parameters to that of the trained personalized model and re-invoke the transfer learning process with more data to update the model's parameters. The privacy enhancement is only employed during inference and does not interfere with the training of the model.

The cloud-based initial training of the general model is performed on a server with a NVIDIA Titan-X GPU with 64GB memory with the same setup as mentioned in Section IV-A.

For device-based personalization, we compare four methods on personal user data:

1) **Reuse:** reusing the general model without modifications to do personalized predictions (baseline)
2) **LSTM:** training a 1-layer LSTM with dropout[1]
3) **TL Feature Extract (FE):** employing transfer learning-based feature extraction on the general model
4) **TL Fine Tune (FT):** employing transfer learning-based fine tuning on the general model

As before, we train individual personalized models for 100 unique and distinct users on a low-end CentOS Linux 7 machine with a 2.20GHz Intel CPU and 8GB RAM. The computing power mimics a resource-constrained mobile device. All personalized models perform grid search using 3-fold time-series cross validation for hyperparameter selection.

[1]Since there is limited personal user training data, a single layer LSTM model with dropout is a sufficient baseline.

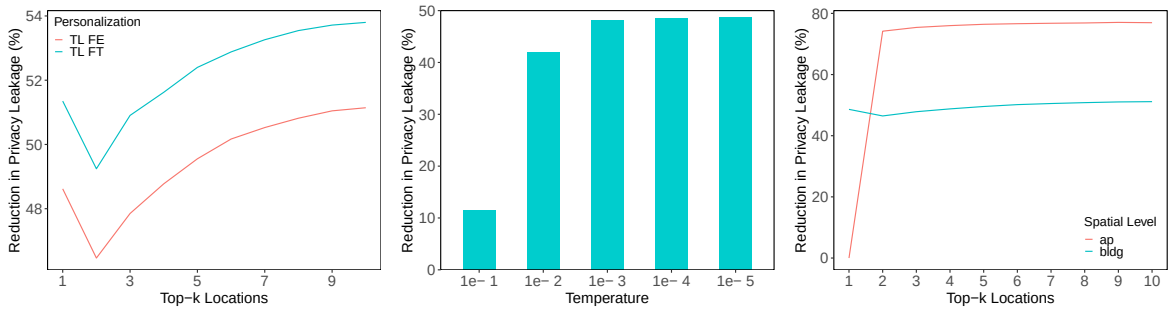
TABLE III: Aggregate train and test accuracy (%) of different personalization methods on 100 individual users. Results demonstrate that transfer learning-based personalization methods employed in *Pelican* increase test accuracy where the feature extraction (FE) method is least prone to overfitting.

Location	Method	Train (%)	Test (%)		
			top-1	top-2	top-3
Building	Reuse	52.16	53.02	60.09	63.68
	LSTM	70.26	60.00	72.03	78.62
	TL FE	67.81	**61.19**	**72.62**	**79.05**
	TL FT	76.47	**60.70**	**73.16**	**79.61**
AP	Reuse	27.02	28.01	32.18	34.42
	LSTM	51.39	44.35	57.60	63.36
	TL FE	60.56	**48.45**	**61.94**	**66.52**
	TL FT	68.38	**47.91**	**62.26**	**67.36**

2) Overhead of Model Personalization: We compare the overheads of the cloud-based initial training and the device-based personalization phases in *Pelican* with the goal that the latter is much less compute intensive than the former since it runs on mobile devices. Our results demonstrate general model training uses approximately 43,000 billion CPU cycles and takes 4.55 hours, whereas personalized modeling uses on average 15 and 14 billion CPU cycles and takes 6.62 and 5.92 seconds for TL FE and TL FT personalization methods respectively (aggregated for 100 users). These results show that while the general model training requires cloud servers, personalization can be done on low-end mobile or edge devices.

3) Efficacy of Device-based Personalization: Table III contains the aggregate results of the personalization methods at building and AP-level locations for 100 distinct users. The reuse method performs the worst in both cases. From the results, we can conclude that the TL FE method performs the best by almost doubling the baseline accuracy for AP predictions and being less prone to overfitting to the personal data compared to the LSTM and TL FT methods. We define overfitting as the discrepancy between train and test accuracy.

The personalized models in Table III are trained with 8 weeks of personal data (note this is equivalent doing device-based personalization followed by iterative model updates in *Pelican*). We further examine the efficacy of *Pelican* with



(a) (b) (c)

Fig. 5: Results of the proposed privacy enhancement: (a) impact of the privacy enhancement on personalized models; (b) impact of varying the privacy parameter; and (c) impact of the privacy enhancement on spatial levels.

TABLE IV: Aggregate train and test accuracy (%) of 100 individual users with different training data sizes. Results demonstrate that transfer learning-based personalization methods employed in *Pelican* are efficient even with less training data whereas the LSTM method is highly prone to overfitting.

Train Data Length	Method	Train (%)	Test (%)		
			top-1	top-2	top-3
2 weeks	LSTM	86.76	46.92	59.67	67.44
	TL FE	67.72	**49.87**	**61.45**	**67.96**
	TL FT	72.99	**51.29**	**61.71**	**68.97**
4 weeks	LSTM	91.63	52.16	65.57	72.73
	TL FE	68.90	**56.64**	**68.16**	**74.97**
	TL FT	78.41	**56.83**	**69.85**	**76.34**
6 weeks	LSTM	91.79	54.12	67.34	74.13
	TL FE	69.03	**58.34**	**70.11**	**76.72**
	TL FT	77.73	**58.90**	**71.90**	**78.37**
8 weeks	LSTM	70.26	60.00	72.03	78.62
	TL FE	67.81	**61.19**	**72.62**	**79.05**
	TL FT	76.47	**60.70**	**73.16**	**79.61**

differing training data sizes. As mentioned earlier, one of the advantages of the transfer learning-based approaches employed in *Pelican* is the ability to use small amounts of training data for learning personalized models. The results of training with differing training data sizes are shown in Table IV for building-level locations. Both the transfer learning personalization approaches perform similarly with only a slight degrade in performance with smaller training data sizes. However, the TL FT and LSTM methods are prone to overfitting with a higher impact on the LSTM performance.

These results also reinforce the complexity of mobility applications [10], [11]. Predicting mobility is difficult and varies by the range of user mobility and correlation between mobility patterns.

4) Privacy Leakage: We perform an evaluation on the reduction in privacy leakage by applying the enhancements presented in Section V-B during attacks for the same set of users in Section IV-B. Without loss of generality, all experiments are performed on adversary A1 using the TL FE personalization method and true p unless otherwise stated. All reported reduction in leakages are aggregated over 100 users.

Impact of privacy layer on personalized models. Results in Figure 5a show the impact of the attack for transfer learning-based personalization methods. The proposed solution is able to reduce privacy leakage by 46-54% for transfer learning methods. The reduction in privacy leakage is higher for transfer learning-based fine tuning and decreases as k increases in both types of models. Since the confidence of the most probable location tends to 1 with the privacy enhancement, the attack becomes solely dependent on the prior information for $k = 1$. Thus, the reduction in privacy leakage is higher for $k = 1$ before decreasing slightly for $k = 2$ and increasing again.

Impact of varying the privacy parameter. Results in Figure 5b demonstrate the impact of changing the temperature (privacy parameter) during inference. As the temperature decreases, the privacy leakage decreases eventually flattening out. Note, this will differ for each user and spatial scales.

Impact of spatial level. Figure 5c contains the results of applying the proposed defense mechanism on different spatial levels. As can be noted, the reduction in privacy leakage is higher for low-level spatial scales than high-level spatial scales for $k > 1$. For the top-1 prediction, the reduction in privacy leakage is bounded at 0.

Key Takeaways: *Pelican* is able to thwart privacy attacks in personalized models with up to 75.41% reduction in leakage while achieving state-of-the-art performance. The privacy enhancement offers a user-centric design to allow users to control the degree of privacy and lowers the ability of the attack to the extent that it is incomprehensible ($< 40\%$ attack efficacy for top-5 predictions) without compromising on model accuracy.

VI. Related Work

Prior defenses against model inversion attacks have been limited and problem specific [20], [27], [28], [37]. Zhao et al. presented a general attribute obfuscation framework using adversarial representation learning to protect sensitive attributes [28]. Yang et al. recently proposed an autoencoder-based prediction purification system to defend against model inversion attacks by minimizing the dispersion in output confidence scores [37]. The purifier is trained by minimizing the inversion attack accuracy and does not coincide with model training. Other defenses that have been proposed to prevent membership

TABLE V: Prior work relevant to defending against attribute-inference attacks.

Phase	Category	Edge Friendly	Model I/O Accessible	Personalized Protection
Data Processing	Artificial data [25], [26]	-	✓	✓
	Data obfuscation [27]–[29]	✓	-	✓
	Light-weight encryption [30]	✓	-	✓
Training	Distributed training [20], [31], [32]	✓	-	✓
	Secure enclaves [33], [34]	-	✓	✓
	Differential privacy perturbation [35]	✓	✓	-
Inference	Output perturbation [17], [36], [37]	-	✓	✓
	***Pelican* (this paper)**	✓	✓	✓

inference attacks may be relevant to model inversion attacks as well. We summarize these in Table V.

Existing defense methods that require changes to the data, such as data obfuscation [28], [29] or encryption [30], do not apply in this application since the output needs to be accessible to the honest-but-curious service provider. Additionally, existing differential privacy-based solutions only apply to multi-user models. In this work, we focus on post-hoc privacy preserving methods that are independent of the trained personalized models. Prior work in this domain [36], [37] induce additional complexity of training noise induction models and are less feasible in applications where the model is on a resource-constrained mobile device.

VII. Conclusion

In this work, we examined the privacy implications of personalized models in distributed mobile services by proposing time-series based model inversion attacks. Our results demonstrated that such attacks can be used to recover historical mobility patterns that may be considered private by the user. We proposed a distributed framework, *Pelican*, that learns and deploys transfer learning-based personalized ML models in a privacy preserving manner on resource-constrained mobile devices. In *Pelican*, we introduced a novel privacy enhancement to thwart model inversion attacks. Our evaluation of *Pelican* using real world traces for location-aware mobile services showed that *Pelican* reduces privacy leakage substantially.

Acknowledgment

We thank the anonymous reviewers for helpful comments. We also thank Priyanka Mammen, Amee Trivedi, Meet Vadera and Walid Hanafy for their feedback. This research was supported in part by NSF grants 1836752, 1763834, Army Research Lab contract W911NF-17-2-0196 and the United States Air Force under contract no. FA8750-17-C-0120. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] P. Zhao, A. Luo, Y. Liu, F. Zhuang, J. Xu, Z. Li, V. S. Sheng, and X. Zhou, "Where to go next: A spatio-temporal gated network for next poi recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[2] X. Dai, I. Spasić, B. Meyer, S. Chapman, and F. Andres, "Machine learning on mobile: An on-device inference app for skin cancer detection," in *IEEE International Conference on Fog and Mobile Edge Computing*, 2019.

[3] S. Yoon, H. Yun, Y. Kim, G.-t. Park, and K. Jung, "Efficient transfer learning schemes for personalized language modeling using recurrent neural network," *AAAI Workshop on Crowdsourcing, Deep Learning, and Artificial Intelligence Agents*, 2017.

[4] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[5] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, "Next place prediction using mobility Markov chains," in *First Workshop on Measurement, Privacy, and Mobility*, 2012.

[6] W. Mathew, R. Raposo, and B. Martins, "Predicting future locations with hidden Markov models," in *ACM Conference on Ubiquitous Computing*, 2012.

[7] X. Song, H. Kanasugi, and R. Shibasaki, "Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level," in *International Joint Conference on Artificial Intelligence*, 2016.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

[9] D. Kong and F. Wu, "HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction," in *International Joint Conferences on Artificial Intelligence*, 2018.

[10] A. Trivedi, J. Gummeson, and P. Shenoy, "Empirical characterization of mobility of multi-device internet users," *arXiv preprint arXiv:2003.08512*, 2020.

[11] J. Feng, C. Rong, F. Sun, D. Guo, and Y. Li, "PMF: A privacy-preserving human mobility prediction framework via federated learning," *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2020.

[12] C. Vallon, Z. Ercan, A. Carvalho, and F. Borrelli, "A machine learning approach for personalized autonomous lane change initiation and control," in *IEEE Intelligent Vehicles Symposium*, 2017.

[13] O. Rudovic, J. Lee, M. Dai, B. Schuller, and R. W. Picard, "Personalized machine learning for robot perception of affect and engagement in autism therapy," *Science Robotics*, 2018.

[14] I. H. Sarker, A. Kayes, and P. Watters, "Effectiveness analysis of machine learning classification models for predicting personalized context-aware smartphone usage," *Journal of Big Data*, 2019.

[15] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symposium on Security and Privacy*, 2017.

[16] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *USENIX Security Symposium*, 2014.

[17] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton, "A methodology for formalizing model-inversion attacks," in *IEEE Computer Security Foundations Symposium*, 2016.

[18] S. Hidano, T. Murakami, S. Katsumata, S. Kiyomoto, and G. Hanaoka, "Model inversion attacks for prediction systems: Without knowledge of non-sensitive attributes," in *International Conference on Privacy, Security and Trust*, 2017.

[19] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment," in *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[20] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Computer Security Applications Conference*, 2019.

[21] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: generative model-inversion attacks against deep neural networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[22] H. Zhang, H. Wu, W. Sun, and B. Zheng, "DEEPTRAVEL: a neural network based travel time estimation model with auxiliary supervision," *International Joint Conference on Artificial Intelligence*, 2018.

[23] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International Conference on Artificial Neural Networks*, 2018.

[24] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, 1999.

[25] A. Triastcyn and B. Faltings, "Generating artificial data for private deep learning," *PAL: Privacy-Enhancing Artificial Intelligence and Language Technologies*, 2019.

[26] X. Zhang, S. Ji, and T. Wang, "Differentially private releasing via deep generative model (technical report)," *arXiv preprint arXiv:1801.01594*, 2018.

[27] K. J. Reza, M. Z. Islam, and V. Estivill-Castro, "Privacy preservation of social network users against attribute inference attacks via malicious data mining," in *International Conference on Information Systems Security and Privacy (ICISSP)*, 2019.

[28] H. Zhao, J. Chi, Y. Tian, and G. J. Gordon, "Trade-offs and guarantees of adversarial representation learning for information obfuscation," *Neural Information Processing Systems (NeurIPS)*, 2020.

[29] T. Zhang, Z. He, and R. B. Lee, "Privacy-preserving machine learning through data obfuscation," *arXiv preprint arXiv:1807.01860*, 2018.

[30] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Network and Distributed System Security Symposium*, 2015.

[31] J. Hamm, A. C. Champion, G. Chen, M. Belkin, and D. Xuan, "Crowd-ML: A privacy-preserving learning framework for a crowd of smart devices," in *International Conference on Distributed Computing Systems*, 2015.

[32] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[33] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving machine learning as a service," *arXiv preprint arXiv:1803.05961*, 2018.

[34] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *USENIX Security Symposium*, 2016.

[35] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[36] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[37] Z. Yang, B. Shao, B. Xuan, E.-C. Chang, and F. Zhang, "Defending model inversion and membership inference attacks via prediction purification," *arXiv preprint arXiv:2005.03915*, 2020.