Generating Daily Activities with Need Dynamics

YUAN YUAN, JINGTAO DING, HUANDONG WANG, and DEPENG JIN, Department of Electronic Engineering, Tsinghua University, China

Daily activity data recording individuals' various activities in daily life are widely used in many applications such as activity scheduling, activity recommendation, and policymaking. Though with high value, its accessibility is limited due to high collection costs and potential privacy issues. Therefore, simulating human activities to produce massive high-quality data is of great importance. However, existing solutions, including rule-based methods with simplified behavior assumptions and data-driven methods directly fitting real-world data, both cannot fully qualify for matching reality. In this article, motivated by the classic psychological theory, Maslow's need theory describing human motivation, we propose a knowledge-driven simulation framework based on generative adversarial imitation learning. Our core idea is to model the evolution of human needs as the underlying mechanism that drives activity generation in the simulation model. Specifically, a hierarchical model structure that disentangles different need levels and the use of neural stochastic differential equations successfully capture the piecewise-continuous characteristics of need dynamics. Extensive experiments demonstrate that our framework outperforms the state-of-the-art baselines regarding data fidelity and utility. We also present the insightful interpretability of the need modeling. Moreover, privacy preservation evaluations validate that the generated data does not leak individual privacy. The code is available at https://github.com/tsinghua-fib-lab/Activity-Simulation-SAND.

CCS Concepts: • **Computing methodologies** → **Modeling and simulation**; **Simulation types and techniques**; **Discrete-event simulation;**

Additional Key Words and Phrases: Daily activities, generation, need dynamics, GAIL

ACM Reference format:
Yuan Yuan, Jingtao Ding, Huandong Wang, and Depeng Jin. 2024. Generating Daily Activities with Need Dynamics. *ACM Trans. Intell. Syst. Technol.* 15, 2, Article 29 (February 2024), 28 pages.
https://doi.org/10.1145/3637493

This work was supported in part by the National Key Research and Development Program of China under grant 2020YFA0711403, the National Nature Science Foundation of China under U22B2057, 62171260, and U20B2060, and the Young Elite Scientists Sponsorship Program by CIC (Grant No. 2021QNRC001).

Authors' address: Y. Yuan, J. Ding (Corresponding author), H. Wang, and D. Jin, Department of Electronic Engineering, Tsinghua University, Beijing, China; e-mails: y-yuan20@mails.tsinghua.edu.cn, dingjt15@tsinghua.org.cn, {wanghuandong, jindp}@tsinghua.edu.cn.

2157-6904/2024/02-ART29 $15.00
https://doi.org/10.1145/3637493

1 INTRODUCTION

Web applications such as Yelp[1] and Meituan[2] have greatly improved the quality of people's daily life and, at the same time, make it possible to record fine-grained activity data. For example, as illustrated in Figure 1, the daily life of an individual is usually logged as an activity sequence, i.e., $S = [a_1, a_2, \ldots, a_n]$, where each entry $a_i = (t_i, k_i)$ contains a timestamp $t_i \in \mathbb{R}^+$ and a discrete activity type $k_i \in C$. Mining activity sequences is valuable for both research and industry in modeling user behaviors and supporting a wide range of applications, like activity planning and recommendation [5, 29, 67]. Despite its high value, only a limited scale of such data is open-sourced for third-party researchers due to privacy-related restrictions on data sharing, which largely hinders the development of downstream applications [31, 32]. Therefore, it is crucial to generate artificial data of human activities by simulation, which can reduce reliance on expensive real data and avoid privacy concerns. In this article, we study the problem of personalized user activity simulation that models the individuals' decision process of what activity to perform at what time, and then generates artificial personalized activity data correspondingly. In order to be publicly shared and used as real-world data, the generated data is expected to be dissociated from real data, i.e., without privacy concerns, and meanwhile capable of retaining data fidelity and utility.

Existing solutions to this problem can be classified into two categories, i.e., *rule-based methods* and *data-driven methods*. *Rule-based methods* that simulate for activity scheduling [4, 7, 14, 34] have a basic assumption that activities can be described by predefined rules derived from activity theories such as utility maximization [57]. However, real-world sequences exhibit complex transition patterns between activities with time dependence and high-order correlations, which are difficult to describe with prior simple rules [15]. Therefore, only relying on simplified assumptions makes *rule-based methods* less qualified for modeling real-world activity behaviors. Instead, *data-driven methods* tackle this problem by directly fitting real-world data. A series of sequential generative methods have been developed, from classical probability models, such as Markov models [54], to deep learning models, such as **Recurrent Neural Networks (RNNs)** [17] and **Generative Adversarial Imitation Learning (GAIL)** [20]. Nevertheless, the above models cannot fully capture the temporal dynamics underlying human daily activities due to the unrealistic inductive bias of being time-invariant [55] or discrete updates only at observed time points [37]. Comparatively, daily activities are always irregularly sampled and longer time intervals introduce larger uncertainty between observations [37, 77], which requires a deeper understanding and fine-grained characterization.

More importantly, there exist complex and various patterns in terms of temporal dynamics of different activities, which are hard to discriminate from each other when mixed together. For example, as Figure 2 illustrates, time intervals of going to the "Concert" exhibit totally distinct patterns compared with going to the "Workplace" that is highly similar to "All". Although individuals lead generally regular daily routines, some activities still occur occasionally but cannot be ignored. However, with the overall distribution exhibiting long-tailed characteristics, the coarse-grained learning paradigm of state-of-the-art *data-driven methods* can be easily biased by the uneven distribution and fail to adequately capture unique patterns of each activity. Therefore, to generate faithful data that matches reality, it is better not to solely rely on the observed data that may possibly reveal an overall but misleading activity pattern.

To address the above issues and achieve a realistic simulation, we propose a novel framework informed by psychological theories and integrate activity-related knowledge into the state-of-the-art GAIL method. Our key idea is to highlight the intrinsic drives of activity decisions, namely, **human**

[1] https://www.yelp.com/
[2] https://about.meituan.com/

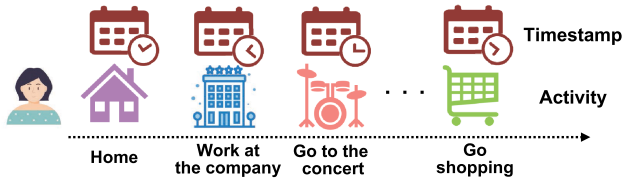


Fig. 1. An example of daily activity sequences, where each entry contains information of the timestamp and activity type.

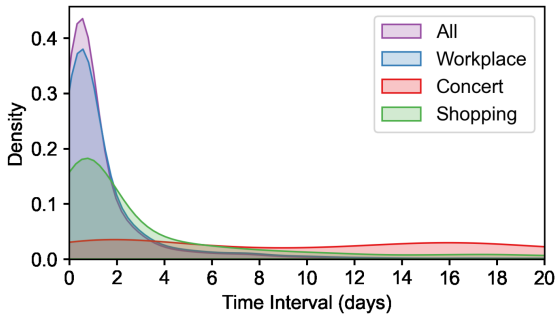


Fig. 2. Interval distributions of different activities. Different activities inherently have distinct temporal dynamics.

needs, which are well supported by Maslow's need theories. Accordingly, human needs can be categorized into three levels: *physiological needs*, *safety needs*, and *social needs*. Guided by this knowledge, we explicitly model human needs in a data-driven manner. We disentangle the needs behind daily activities to fully capture the aforementioned complex patterns in empirical data. Specifically, we simultaneously model each need dynamics with an alternating process between *spontaneous flow* and *instantaneous jump*. For example, the accumulation of needs in evolution (flow) triggers the occurrence of related activities while the decaying needs after satisfaction (jump) can restrain tendencies towards specific activities.

In terms of the specific model design, the proposed GAIL-based framework consists of a discriminator that provides reward signals and a generator that learns to generate high-quality activities with a policy network. Particularly, we utilize Maslow's Theory in our framework to enhance the activity simulation with need modeling from the following two perspectives. First, to overcome the challenge of complex activity patterns, we design a hierarchical structure in the modeling to disentangle different need levels and explicitly incorporate the underlying influence of human needs on activity decisions. Second, to address the limitations of RNN-based methods in modeling continuous-time dynamics, we leverage Neural Stochastic Differential Equations [26] to capture piecewise-continuous characteristics of need dynamics alternating between *spontaneous flow* and *instantaneous jump*. The above need dynamics further serve as the states that define the policy function, which calculates activity intensities based on the current need state and decides the next action accordingly. In conclusion, our contributions can be summarized as follows:

— We are the first to explicitly model the intrinsic drives of activities, i.e., human needs, which brings the synergy of psychological theories and data-driven learning;
— We propose a novel knowledge-driven activity simulation framework based on GAIL, leveraging Maslow's theory to enhance the simulation reality by capturing need dynamics;
— Extensive experiments on two real-world datasets show the effectiveness of the framework in generating synthetic data regarding fidelity, utility, and interpretability.

— We conduct comprehensive evaluations regarding privacy preservation, including uniqueness testing, membership inference attacks, and differential privacy, which demonstrate that the generated data does not leak user privacy.

Compared with the preliminary version [77], the following fields are substantially enhanced. First, privacy issues are crucial for generating user-related data. In this article, we add a comprehensive evaluation of user privacy preservation in Section 4.8, including uniqueness testing, membership inference attacks, and differential privacy. Second, to illustrate the model's ability in addressing the problem of uneven activity distributions, we add an extra study to investigate the model's performance regarding fine-grained metrics in Section 4.4. Third, to demonstrate the model's performance more intuitively and clearly, we add experimental results on a synthetic dataset in Section 4.7, where the ground-truth needs are known and follow a certain pattern.

2 PRELIMINARIES

2.1 Problem Statement

Daily activity data can be defined as a temporal sequence of events $S = [a_1, a_2, \ldots, a_n]$, where a_i is a tuple (t_i, k_i), t_i denotes the timestamp and k_i is the activity type, e.g., eating at restaurants, working at companies, playing at sports centers. The problem of activity simulation can be defined as follows:

Definition 1 (Human Activity Simulation). Given a real-world activity dataset, generate a realistic activity sequence $\hat{S} = [\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_n]$ with a parameterized generative model.

2.2 Temporal Point Process

A **temporal point process (TPP)** [44] can be realized by an event sequence $\mathcal{H}_T = \{(t_1, m_1), \ldots, (t_n, m_n) | t_n < T\}$. Here t_i represents the arrival time of the event and m_i is the event mark. Let $\mathcal{H}_t$ denote the history of past events up to time t, the conditional intensity function $\lambda_k^*(t)$ (the kth event category) is defined as:

$$\lambda_k^*(t) = \lim_{\Delta t \to 0^+} \frac{\mathbb{P}(\text{event of type} k \text{ in} [t, t + \Delta t] | \mathcal{H}_t)}{\Delta t}. \tag{1}$$

Note that $\lambda^*(t) = \sum \lambda_k^*(t)$ denotes the total conditional intensity, deciding the arrival time without considering event types. Then, the event type is sampled at the probability proportional to $\lambda_k^*(t)$.

2.3 Neural Ordinary Differential Equations

Neural Ordinary Differential Equations (NODE) is a continuous-time neural network [10], which describes the evolution of the latent state $\mathbf{z}(t)$ over continuous time $t \in \mathbb{R}$. Specifically, the dynamics of systems can be captured by modeling the first-order **ordinary differential equations (ODE)** with neural networks. NODE has shown plausible performance in modeling dynamic systems [12, 58, 60]. Specifically, the derivative of the latent state is modeled as:

$$d\mathbf{h}(t) = f(\mathbf{h}(t), t; \theta) \cdot dt \,, \tag{2}$$

where $\mathbf{h}(t)$ is the representation of the latent state at time t and f describes the derivative at time t, which is parameterized by a neural network. In this way, the output of the system at time t_1 can be solved with an initial value at time t_0 by an ODE solver:

$$\mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t; \theta) \cdot dt \,, \tag{3}$$

In this work, we take the first attempt to characterize human needs with neural differential equations.

Table 1. The Description of Commonly Used Notations

Notations	Descriptions
MDP	Markov decision process
$< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} >$	State space, action space, state transition and reward function.
$\mathbf{z}(t) = \{\mathbf{z}_1(t), \mathbf{z}_2(t), \mathbf{z}_3(t)\}$	Hierarchical need embedding process
$\mathcal{F}, \mathcal{G}$	Functions of Spontaneous flow and instantaneous jump
$\mathbf{c}_i(t), \mathbf{h}_i(t)$	Internal embedding process and historical embedding process
$\lambda_k(t)$	Intensity function for the kth activity type

3 METHOD

We first introduce how we model human needs to motivate the framework design in Section 3.1, then explain the Markov decision process modeling of the decision process in Section 3.2, and finally elaborate on the framework details in Section 3.3. We also explain the commonly used notations in Table 1.

3.1 Human Needs Modeling

Hierarchy of Needs. According to a classic theory in psychology, i.e., Maslow's Theory [43], people are motivated to achieve a hierarchy of needs, including *physiological needs*, *safety needs*, *social needs*, *esteem needs*, and *self-actualization needs*, in a priority order, where higher levels of need are modeled as long-term changes such as life stages. With the development of Maslow's Theory, the follow-up theories [8, 11, 56] have introduced flexibility in the hierarchy. For example, different needs can be pursued simultaneously, and there exist transition probabilities between any pair of needs. We do not take the top two need levels for *esteem* and *self-actualization* into consideration because they are too abstract and their effects can only be observed in a long term.

Here we classify individuals' activities into three need levels, including *physiological needs* (level-1), *safety needs* (level-2), and *social needs* (level-3), which are sufficient to depict patterns of daily life [30, 31]. These three need levels are often triggered or satisfied in a short period, which are consistent with daily activities that happen within a short term (a few hours). We provide descriptions of each need level as follows:

— ***Physiological needs*** refer to biological requirements for survival, e.g., food, drink, and shelter. The human body cannot function optimally without satisfying these needs.

— ***Safety needs*** refer to requirements for security and safety, e.g., education and employment. Besides physiological needs, people expect their lives to be orderly, regular, and controllable.

— ***Social needs*** refer to requirements for spirits, e.g., entertainment and social relationships. After meeting physiological and safety needs, people are also striving for spiritual satisfaction.

In our modeling, we follow Maslow's Theory in a more flexible way, rather than the original needs pursued in a rigid order. The fulfillment order can be flexible according to individual preferences and external circumstances. Based on well-respected need theories, each activity is explicitly labeled with one of the need levels.[3] The association between human needs and activities based on expert knowledge bridges the gap between classic psychological theories and human behavior modeling, which provides opportunities to model human needs computationally in a data-driven manner.

Evolution of Needs. In real-world scenarios, human needs are not static but generally evolve with time dynamically, which not only derive from spontaneous changes, but also can be interrupted by happened activities. To better learn sequential activity patterns, it is essential to capture the

[3]We refer the readers to Section 4.1.2 for more details of the need annotation.

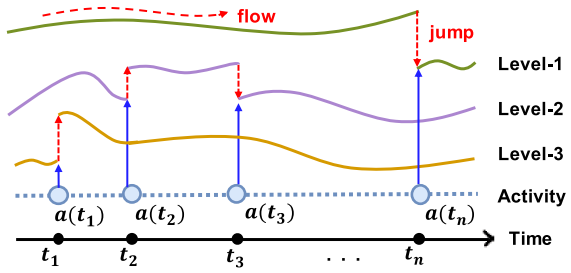


Fig. 3. Illustration of the need evolution. Representations of three-level need states evolve continuously over time until interrupted at the time when a corresponding activity happens (e.g., $a(t_1)$ corresponds to level-1). Note that the need state is modeled by an embedding rather than a scalar, thus the jumps up and down do not indicate an increase or decrease.

underlying mechanism of need dynamics. However, it is non-trivial because human needs cannot be observed explicitly and are affected by various factors, such as activity relations and periodicity. Besides, different from activities that happen one by one, need dynamics are more complicated with synchronicity and competitiveness among different levels.

To effectively capture the underlying need dynamics, we innovatively capture piecewise-continuous dynamics in human needs including *spontaneous flow* and *instantaneous jump* as follows:

- ***Spontaneous flow*** denotes the continuous-time flow of need states. For example, needs for some activities can accumulate without taking them for a long time. Meanwhile, needs can also decay gradually as time goes by.
- ***Instantaneous jump*** models the influence of activities on the need states. For instance, the happened activities can immediately change the evolutionary trajectory of the corresponding need state.

Naturally, the two kinds of dynamics describe an active process of need evolution and need satisfaction.

Particularly, the three levels are disentangled in dynamic modeling, so they follow distinct evolution laws. Figure 3 illustrates the two evolution mechanisms of different need levels. Nevertheless, it is challenging to learn such dynamics since needs are intrinsically unobserved and stochastic with the coexistence of continuity and jump. To tackle this problem, we represent human needs with a stochastic embedding process $\mathbf{z}(t)$ defined as follows:

Definition 2 (Need Embedding Process). The need embedding processes are $\{\mathbf{z}_i(t), i \in \{1, 2, 3\}, t \geq 0\}$, where $\mathbf{z}_i(t)$ is the representation of the ith need level at time t.

In the above definition, we depict human needs with an embedding process $\mathbf{z}(t)$ instead of a direct scalar value for stronger representation capabilities. Particularly, $\mathbf{z}(t)$ is composed of three components $\mathbf{z}_1(t)$, $\mathbf{z}_2(t)$, $\mathbf{z}_3(t)$ that correspond to different need levels. Then the need embedding process $\mathbf{z}(t)$ with both *spontaneous flow* and *instantaneous jump* can be formulated as follows:

$$\begin{cases} \mathbf{z}(t+dt) = \mathbf{z}(t) + \mathcal{F}(t, \mathbf{z}(t))dt, & \text{no activity in } [t, t+dt), \\ \lim\limits_{\Delta t \to 0^+} \mathbf{z}(t_i + \Delta t) = \mathcal{G}(t_i, \mathbf{z}(t_i), k(t_i)), & \text{with activity } k \text{ at } t_i, \end{cases} \tag{4}$$

where $\mathcal{F}$ and $\mathcal{G}$[4] control the *spontaneous flow* and *instantaneous jump*, respectively, and $k(t_i)$ denotes the the occurred activity.

[4]The time-dependent variables in our modeling are all left continuous in t, i.e., $\lim\limits_{\epsilon \to 0^+} \mathbf{z}(t-\epsilon) = \mathbf{z}(t)$.

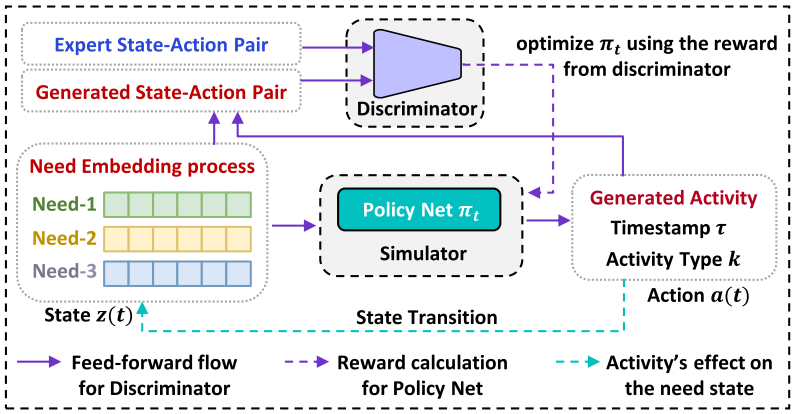


Fig. 4. Illustration of the SAND framework. The policy and discriminator networks are optimized adversarially, and the state transition consists of two evolution mechanisms.

3.2 Sequential Decision Processes

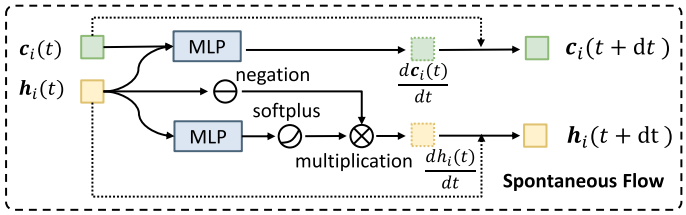
The generation of activity sequences depends on individuals' decisions on what activity to take based on his/her own need state step by step. The whole process consists of a sequence of activity decisions that aim to maximize the total received "reward" along the process. Here we model the decision process as a Markov decision process (MDP) [65], and it is described by a 4-tuple $< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} >$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is the state transition, and $\mathcal{R}$ is reward function. The basic elements of MDPs are: (i) **State** represents the current need state; (ii) **Action** is generated based on the state by sampling a time interval τ and an activity type k; (iii) **Policy function** decides the next activity time and type; (iv) **State transition** controls how the state updates with two transit laws, i.e., spontaneous flow and instantaneous jump; and (v) **Reward function** evaluates the utility of taking the action under the state, which is unknown and has to be learned from the data.

Given the activity history $\mathbf{s}_t = \{(t_i, k_i)\}_{t_i<t}$, the stochastic policy function $\pi_\theta(a|\mathbf{s}_t)$ samples an interval time τ and an activity type k to generate the next activity $a = (t_{i+1}, k_{i+1})$, where $t_{i+1} = t_i + \tau$. Then, a reward value is calculated and the state will be updated by an *instantaneous jump*. Besides, there are also feedbacks of need states to the individual over time known as the *spontaneous flow*.

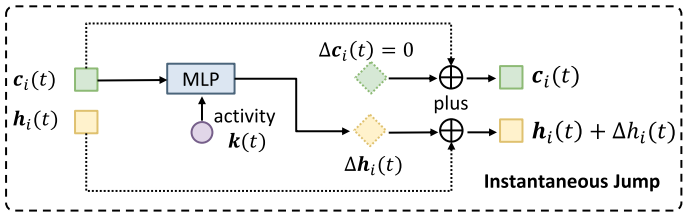
3.3 Proposed Framework: SAND

In this section, we present a novel framework, SAND, which **S**imulates human **A**ctivities with **N**eed **D**ynamics. Overall, it provides the synergy of need theories and imitation learning in simulating the activity decision-making process. As shown in Figure 4, it learns the policy and reward functions adversarially, where the need embedding process $\mathbf{z}(t)$ plays an essential role in the loop. We elaborate on the details of key components in the following sections.

3.3.1 Learning Need Dynamics. To model the need dynamics including the *spontaneous flow* and *instantaneous jump*, we utilize neural stochastic differential equations [26] to describe such continuity and discontinuity, where the need embedding process $\{\mathbf{z}_i(t), i \in \{1, 2, 3\}, t \geq 0\}$ acts as the latent state. Between activity observations, each $\mathbf{z}_i(t)$ flows continuously over time. Once an activity happens, the corresponding need embedding process is interrupted by a state jump. Different from directly modeling the changes of the hidden state like RNNs [69], neural differential equations model the derivative of $\mathbf{z}(t)$ to better capture the continuous-time characteristics.



(a) Spontaneous flow



(b) Instantaneous jump

Fig. 5. Network architecture to learn need dynamics with both spontaneous flow and instantaneous jump. (a) shows the spontaneous flow of $\mathbf{c}_i(t)$ and $\mathbf{h}_i(t)$ based on the derivatives $\frac{d\mathbf{c}_i(t)}{dt}$ and $\frac{d\mathbf{h}_i(t)}{dt}$. (b) illustrates the instantaneous jump caused by the happened activity k.

Specifically, the derivative of the ith need state is formulated as follows:

$$d\mathbf{z}_i(t) = f_i(\mathbf{z}_i(t), t; \theta_i) \cdot dt + \omega_i(\mathbf{z}_i(t), \mathbf{k}_i(t), t; \gamma_i) \cdot dN_i(t), \tag{5}$$

where f_i and ω_i are both parameterized by neural networks and control the *spontaneous flow* and *instantaneous jump* of the ith need embedding process, respectively, and $N_i(t)$ records the number of activities of the ith level up to time t. f and ω in Equation (5) are implementations of the function $\mathcal{F}$ and $\mathcal{G}$ defined in Equation (4). In particular, each state $\mathbf{z}_i(t) \in \mathbb{R}^n$ is composed of two vectors: (1) $\mathbf{c}_i(t) \in \mathbb{R}^{n_1}$ encodes the internal need state, and (2) $\mathbf{h}_i(t) \in \mathbb{R}^{n_2}$ encodes effects of the historical activities.

Spontaneous Flow. The top part in Figure 5 shows the network design to model spontaneous flow. The neural function f_i in Equation (5) controls the spontaneous flow of the state $\mathbf{z}_i(t)$. Although $\mathbf{z}_i(t)$ contains two vectors $\mathbf{c}_i(t)$ and $\mathbf{h}_i(t)$, they follow distinct continuous dynamics due to different encoded information. Specifically, there is no constraint on the internal evolution of $\mathbf{c}_i(t)$, hence we model $\frac{d\mathbf{c}_i(t)}{dt}$ by an MLP. On the other hand, due to the temporal decaying effect of historical activities, we add constraints to the form of $\mathbf{h}_i(t)$ to model such an effect. Concretely, we use another MLP followed by a Softplus activation layer to model the decay rate. The modeling of derivatives can be formulated as follows:

$$\frac{d\mathbf{c}_i(t)}{dt} = \text{MLP}(\mathbf{c}_i(t) \oplus \mathbf{h}_i(t)), \tag{6}$$

$$\alpha_i = \sigma(\text{MLP}(\mathbf{c}_i(t)), \tag{7}$$

$$\frac{d\mathbf{h}_i(t)}{dt} = -\alpha_i \mathbf{h}_i(t), \tag{8}$$

where σ is the Softplus activation function to guarantee a positive decay rate, and $\oplus$ denotes the vector concatenation.

Instantaneous Jump. The bottom part in Figure 5 illustrates the network design to model the instantaneous jump introduced by happened activities. Specifically, the function ω_i in Equation (5)

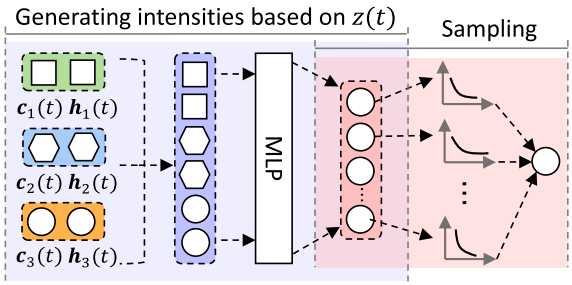


Fig. 6. Network architecture of the policy function.

outputs the effects of the instantaneous jump, and it is modeled by an MLP in practice. As discussed before, the vector $\mathbf{h}_i(t)$ encodes the activity memory, and thus it is reasonable that the instantaneous jump will only affect the vector $\mathbf{h}_i(t)$. As a result, an activity of the i_{th} need level gives rise to a change $\Delta\mathbf{h}_i(t)$ only to the corresponding activity memory embedding $\mathbf{h}_i(t)$, i.e., $\Delta\mathbf{h}_j(t) = 0, \forall j \neq i$ and $\Delta\mathbf{c}_i(t) = 0, \forall i$. The MLP takes in the concatenation of the activity embedding $\mathbf{k}(t)$ and the internal state $\mathbf{c}_i(t)$, and outputs the variation $\Delta\mathbf{h}_i(t)$ in the memory embedding $\mathbf{h}_i(t)$, which is formulated as follows:

$$\Delta\mathbf{h}_i(t) = \text{MLP}(\mathbf{k}(t) \oplus \mathbf{c}_i(t)), \tag{9}$$

$$\lim_{\epsilon \to 0^+} \mathbf{h}_i(t+\epsilon) = \mathbf{h}_i(t) + \Delta\mathbf{h}_i(t), \tag{10}$$

where $\mathbf{k}(t)$ denotes the activity associated with the ith need level.

3.3.2 Policy Function. Based on the activity intensity function $\lambda_k(t)$, the probability of activity type k happens within the time interval $[t, t+dt)$ is as: $P\{\text{activity } k \text{ happens in}[t, t+dt)\} = \lambda_k(t) \cdot dt$. The policy function is a mapping from the state to action that generates the arrival of the next activity with the type conditioned on the current state. With the modeling of activity intensities, the goal of the policy function is to generate intensities based on the need states $\mathbf{z}(t)$. Figure 6 shows the network design of the policy function. Although the three need levels control specific activities, they are not independent and can be pursued simultaneously, which may give rise to competing activity choices. Therefore, the states of the three levels all affect the generation of the next activity. In other words, the activity intensity $\lambda_k^*(t)$ is conditioned on embedding processes of all need levels. To model the interactions between different levels in determining the next activity, we concatenate the three embedding processes $\mathbf{z}_i(t), i \in \{1, 2, 3\}$ and leverage an MLP to obtain conditional activity intensities. Here we perform the sampling to obtain the time interval and the activity type based on the total condition intensity and type distribution as:

$$\lambda^*(t) = \sum_{k=1}^{M} \lambda_k(t), \quad p(k|t) = \frac{\lambda_k(t)}{\sum_{k=1}^{M} \lambda_k(t)} \tag{11}$$

where M is the number of activity types.

3.3.3 Reward Function. GAIL uses a reward function to evaluate the actions by comparing the generated state-action pairs with the real pairs, which is modeled by a discriminator network. To compare the real and generated pairs more effectively, we also adopt the historical sequence of activities as part of the state, thus, the state in the discriminator is defined as $s_d = (\mathbf{z}(t), \mathbf{S})$. For the sequence $\mathbf{S}$, we consider the information of time intervals, hour, weekday, activity type, and need. As a result, each element of $\mathbf{S}$ is a 5-tuple $(x_i, h_i, w_i, c_i, n_i)$, where x_i, h_i, w_i, c_i, n_i represents

the time intervals, hour, weekday, activity type, and the need of the ith activity, respectively. In addition, the action a is set as the time interval x since the last activity and the activity type c, i.e., $a = (x, c)$. Based on the above notations, the output of the discriminator can be defined as $D_\phi(s_d, a)$.

Through an embedding layer, we first transform the items in the historical sequence s_d and the action a into embeddings. Then we leverage an attention mechanism to aggregate the sequential features. The concatenation of the sequential embedding, state $\mathbf{z}(t)$ and action embedding is fed into an MLP with a sigmoid activation function. The discriminator $D\phi$ can be formulated as follows:

$$\begin{aligned}
&\mathbf{h}^s = \text{Embedding}(s_d), \mathbf{h}^a = \text{Embedding}(a),\\
&m_j = \mathbf{u}_\alpha^T \tanh(\mathbf{W}_u \mathbf{h}_j^s + \mathbf{b}_u),\\
&\alpha_j = \frac{\exp(m_j)}{\sum_{k=1}^{l} \exp(m_k)},\\
&\mathbf{e}^s = \sum_i \alpha_j \mathbf{h}_j^s,\\
&d = \text{Sigmoid}(\text{MLP}(\mathbf{e}^s \oplus \mathbf{z}(t) \oplus \mathbf{h}^a)),
\end{aligned} \tag{12}$$

where j denotes the jth item in the historical activity sequence, $\mathbf{W}_u$ and $\mathbf{b}_u$ are parameters of the linear layer, and μ_α is a learnable vector. In this way, the reward function can be expressed as follows:

$$R(s, a) = \log D_\phi(s_d, a), \tag{13}$$

3.4 Training and Simulation

3.4.1 Model Training. **GAIL Training.** The training procedure of GAIL is an iterative process of learning a non-linear policy function π_θ and a non-linear discriminator D_ϕ. At each iteration, the policy network generates activity sequences $\mathcal{T}_G$, and then each state-action pair is assigned a reward provided by the discriminator. Then the parameters of the policy network are updated with the reward via the PPO algorithm, which is widely used in reinforcement learning [61]. Subsequently, the generated sequences $\mathcal{T}_G$ and real-world sequences $\mathcal{T}_E$ are used as the training data to optimize discriminator parameters D_ϕ with gradient ascent on the following loss function:

$$\mathcal{L}_D = \mathbb{E}_{(s,a)\in\mathcal{T}_E} \log D_\phi(s, a) + \mathbb{E}_{(s,a)\in\mathcal{T}_G} \log(1 - D_\phi(s, a)). \tag{14}$$

Therefore, the optimization of policy π_θ and discriminator $D\phi$ is to solve a min-max problem with the following objective:

$$\max_\phi \min_\theta -\lambda H(\pi_\theta) + \mathbb{E}_\pi[\log D_\phi(s, a))] + \mathbb{E}_{\pi_E}[\log(1 - D_\phi(s, a))], \tag{15}$$

where $\mathbb{E}_\pi$ represents the expected reward of the sequences under the policy π, and π_E represents the policy under the expert sequences. $H(\pi_\theta)$ is an entropy regularization term, controlling to find the policy π with maximum causal entropy.

Pre-training Mechanism. In order to accelerate the training procedure and improve the performance of the adversarial framework, we first pre-train the embedding network of need dynamics and policy network. In this way, we can enable the generator to preview the important interactions between human needs and activities before GAIL training. To achieve this goal, we utilize the log-likelihood loss function [26] as follows:

$$\mathcal{L} = -\sum_j \log\lambda(\mathbf{z}(\tau_j)) - \sum_j \log p(\mathbf{k}_j|\mathbf{z}(\tau_j)) + \int_{t_0}^{t_N} \lambda(\mathbf{z}(t))dt, \tag{16}$$

Besides, we also pretrain the discriminator, where we randomly generate activity sequences as fake data. With the fake data and real data, the discriminator is trained to distinguish them with a binary classification loss.

3.4.2 Activity Simulation. After training the overall framework, we are able to perform activity simulation. First, the states for the three need levels are randomly initialized. Then, the continuous dynamics are calculated based on the learned model parameters until an activity occurs, which also introduces an instantaneous jump to the dynamical evolution. In this way, activities in a sequence can be generated regressively. Algorithm 1 presents the simulation procedure.

ALGORITHM 1: Activity simulation with need dynamics (generating one individual as an example)

Input: Parameter of the policy function π, start time t_0, end time t_n, initial need state $\mathbf{z}_i(t_0), i \in \{1, 2, 3\}$.
Output: Activity sequence H

initialize $t = t_0, H = \{\}, \mathbf{z}_i = \mathbf{z}_i(t_0), i \in \{1, 2, 3\}$
while $t < t_n$ **do**
  dt = AdpativeStepSize($\mathbf{z}_i$, t, π)
  $(t_{\text{next}}, k_{\text{next}})$ = SimulateNextActivity($\mathbf{z}_i$, t, π)
  if $t_{\text{next}} > t + dt$ **then**
    $\mathbf{z}_i$ = SpontaneousFlow($\mathbf{z}_i$, dt, π), $i \in \{1, 2, 3\}$
  else
    $H = H \cup \{(t_{\text{next}}, k_{\text{next}})\}$
    $\mathbf{z}_i$ = SpontaneousFlow($\mathbf{z}_i$; dt; π), $i \in \{1, 2, 3\}$
    $\mathbf{z}_i$ = InstantaneousJump ($\mathbf{z}_i$, $(t_{\text{next}}, k_{\text{next}})$, π), i corresponds to the activity type k_{next}
  end if
  $t = t + dt$
end while

4 EXPERIMENTS

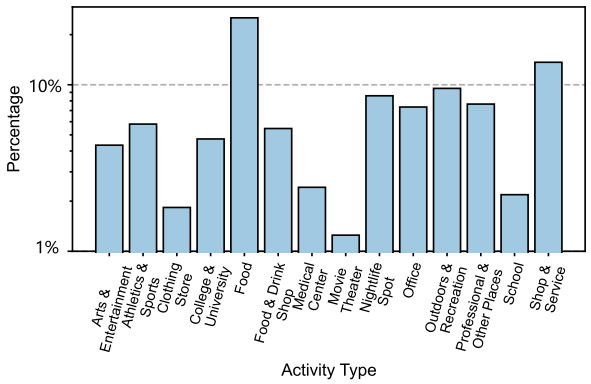
In this section, we conduct extensive experiments to investigate the following research problems:

— **RQ1:** How does SAND perform in retaining the data fidelity and reflecting activity characteristics compared with baseline solutions?
— **RQ2:** How do different components of SAND contribute to the final performance?
— **RQ3:** Can SAND generate high-quality synthetic data that benefit practical applications?
— **RQ4:** Does SAND memorize any identifying information and whether the generated synthetic data leak user privacy?

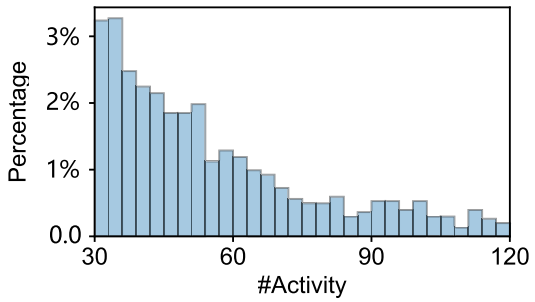
4.1 Experimental Settings

4.1.1 Datasets. We conduct extensive experiments on two real-world datasets:

— **Foursquare-NYC [72].** This dataset contains check-in activities to various POIs collected from 2,000 users with 14 activity labels during the duration from 2012-05-01 to 2012-06-01. We filter out users with less than 30 activities during this period. Each check-in behavior is equivalent to an occurred activity, with the information of the anonymous user ID, timestamp, and venue category. In the same way, we use the venue category as the activity type. Therefore, the activity types in this dataset contain the following categories: Arts and Entertainment, Athletics and Sports, Clothing Store, College and University, Food, Food and Drink Shop, Medical Center, Movie Theater, Nightlife Spot, Office, Outdoors and Recreation, Professional and Other places, School, Shop and Service. Figure 7 shows the statistical distribution.



(a) Distribution of activity type



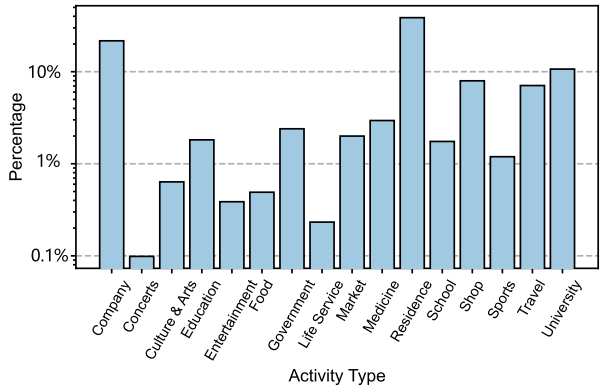
(b) Distribution of the number of activities

Fig. 7. The statistics of the Foursquare dataset. (a) Distribution of activity type with y-axis in log scale. (b) Distribution of the number of activities.

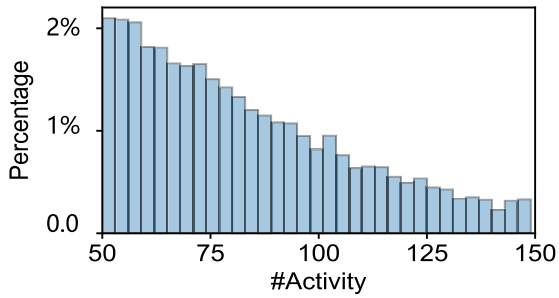
— **Mobile Operator.** This dataset contains 10,000 users with 15 activity labels during the duration from 2016-09-17 to 2016-10-17, which is collected in Beijing by a major mobile operator in China. We filter out users with less than 50 activities in this period. The data records the anonymous user ID, the visited location category, and the timestamp of each visit. As activity data are characterized by visits to different types of locations, we use the location category as the activity type. Specifically, the activity types in this dataset contain the following categories: Company, Concerts, Culture and Art, Education, Entertainment, Food, Government, Life Service, Market, Medicine, School, Shop, Sports, Travel, and University. Figure 8 shows the statistical distribution.

As we can see, individuals do not distribute uniformly across different activities. There are more visits to companies, residences, and schools, which are consistent with daily life patterns. It should be pointed out that the activity of going home (visiting residential venues) is so sparse in the Foursquare data that we do not consider it, which is different from the Mobile Operator Dataset. Actually, flexible characterization of needs with activities also exhibits the generalization ability of the proposed framework to different scenarios.

We take careful steps to consider ethical issues in using data: First, the Terms of Service for both datasets include consent for research studies. Second, the research protocol has been reviewed and approved by our local institutional board. All research data is sanitized for privacy preservation, with limited access to authorized researchers bound by non-disclosure agreements.



(a) Distribution of activity type



(b) Distribution of the number of activities

Fig. 8. The statistics of the mobile operator dataset. (a) Distribution of activity type with y-axis in log scale. (b) Distribution of the number of activities.

4.1.2 Need Annotation. According to the definition and description of each need level in Section 3.1, we ask three annotators to label each activity with one of the need levels. To ensure that correct expert knowledge is utilized, the three annotators all have expertise in related knowledge, including a senior Ph.D. candidate and two postdocs with a background in psychology and behavioral sciences. We choose the number of experts (three) following studies in NLP with annotation tasks [64]. If the three experts disagree on the label, we will invite another expert and start a discussion. Through this process, all activities obtain consistent labels. The annotation approach has satisfied the requirement of our problem settings due to the small scale of activity types. Generally speaking, there are limited activity types in daily life, thus, massively scalable annotation methods are not required in most cases. By leveraging the relationships between needs and activities based on need theories, we classify each activity into one of the three levels, and it is a knowledge-driven process.

Moreover, we provide scalable methods if more fine-grained activities are considered. First, we can ask experts to provide detailed definitions, descriptions, and examples of relationships between activities and need levels. Then, we can use Amazon's Mechanical Turk[5] to obtain a reliable

[5]It is a crowdsourcing website to perform discrete on-demand tasks that computers are currently unable to do: https://www.mturk.com/.

need-level label for each activity. Specifically, we assign a need-level label to an activity if two annotators give consistent labels, and ask another annotator to judge it if annotators disagree on the label. To assess the annotation reliability, we can compute the intra-class correlation [6]. In this way, the utilization of expert knowledge is scalable.

This annotation approach offers potential benefits that extend beyond the scope of our current algorithms. One notable application is in recommendation algorithms. By assigning need-level labels to item categories, guided by psychological theories, we can harness these annotations to capture users' high-level demands, going beyond their low-level item preferences. This approach represents a valuable means of expanding the significance of our research, while also offering valuable insights to the wider research community.

4.1.3 Baselines. To evaluate the performance of the SAND framework, we compare it against state-of-the-art baseline methods:

- *Semi-Markov [38].* This classical probability model defines a state for every given time, not just at the activity jump times, and builds a transition matrix to capture the first-order transition probabilities between activities.
- *Hawkes Process [35].* It is a representative point process model, where the arrival of activity causes the conditional intensity function to increase. All previously occurred activities affect the intensity function explicitly.
- *Neural Hawkes Process [44].* It is the neural extension to the Hawkes process, where the intensities of multiple activities evolve according to a continuous-time LSTM, and thus it allows past activities to affect the future in realistic ways.
- *Transformer Hawkes Process (THP) [82].* It is another neural extension to the Hawkes process, which utilizes the self-attention mechanism to capture long-term dependencies.
- *Neural Jump Stochastic Differential Equations (NJSDE) [10].* This is the state-of-the-art method to learn continuous and discrete dynamic behavior. We use this baseline without the need modeling.
- *LSTM [21].* It is widely used in sequence prediction. Here we combine an attention mechanism with the recurrent network to capture the activity patterns.
- *SeqGAN [74].* It is the state-of-the-art discrete generative model, which combines generative adversarial nets with reinforcement learning.
- *Generative Adversarial Imitation Learning (GAIL) [20].* This is a model-free imitation learning algorithm, which is the state-of-the-art method in imitating complex decision-making processes.

4.1.4 Metrics. We measure whether synthetic data accurately reflects crucial characteristics of the original, real-world data. Following the mainstream practice in previous works [15, 50], we use essential metrics to describe activity patterns for comparing the statistical similarity between the generated data and real-world data, including (1) *ActInt*: Time intervals between activities, including type-free intervals (MacroInt) and type-aware intervals (MicroInt); (2) *DailyAct*: Daily happened activities. It is the number of activities in one day for each individual; (3) *ActType*: The overall distribution over different activity types; (4) *Weekday*: The overall time distribution over the seven days; (5) *Hour*: The overall time distribution over the twenty-four hours. To get the quantitative evaluations on the fidelity of generated data, we use **Jensen–Shannon divergence (*JSD*)** to measure the distribution similarity of the above patterns between the generated data and real-world data, which is a widely used distance metric for comparing two distributions. Specifically, the JSD is defined as follows:

$$\mathrm{JSD}(P||Q) = H(M) - \frac{1}{2}(H(P) + H(Q)), \tag{17}$$

Table 2. Values of Hyper-parameters

Notations	Descriptions
Number of need levels	3
Embedding size of MLPs	32
Embedding size of the need embedding process	24
Batch size	32
Replay buffer size (sequence-level)	128
Learning rate	3×10^{-4}
Entropy coefficient in PPO	0.01
Optimizer	Adam
ODE solver	Jump Adams

where H is the Shannon entropy, p and q are distributions, and $M = \frac{p+q}{2}$. In our setup, lower JSD denotes a closer distribution between synthetic data and real data, which indicates a better generative model. In addition, the JSD is bounded by $[0, 1]$ for two probability distributions with the base 2 logarithm [39].

4.2 Parameter Settings

In our experiments, we all use two-layer MLPs with a hidden size of 32. The embedding size of the vector $\mathbf{c}_i(t)$ and the embedding size vector $\mathbf{h}_i(t)$ are both set as 4. Therefore, the embedding size of the need embedding process $\mathbf{z}(t)$ is 24, i.e., $(4 + 4) \times 3$, which is the concatenation of states associated with different needs. Besides, the simulation is performed in batch, which is set as 32. After training the policy network ten times, we train the discriminator once with the accumulated state-action pairs. Thus, the number of inputs of the discriminator network is ten times of the policy network. In addition, we set the learning rate as 3e-4 via searching in a set of $\{1e-4, 3e-4, 1e-3, 3e-3\}$. Besides, we provide detailed values of the hyper-parameters in Table 2 for reproducibility. The proposed framework is implemented with Pytorch and trained on a Linux server with eight GPUs (NVIDIA GTX 1080 * 8). In practice, our framework can be effectively trained within 8 hours on a single GPU. As for some baselines that use CPUs, the models are trained on a Linux server with two CPUs (Intel Xeon E5-2650 * 2).

4.3 Performance Comparison (RQ1)

4.3.1 Overall Performance. Table 3 reports the performance in retaining the data fidelity of our framework and the eight competitive baselines on two real-world datasets. From the results, we have the following findings:

- **Our framework steadily achieves the best performance.** SAND achieves the best performance on the mobile operator dataset, by ranking first on five metrics and second on one metric. For five metrics that rank 1st, SAND reduces the JSD by more than 20%. It also shows superior performance on most of the metrics on the Foursquare dataset, which ranks first on five metrics by reducing JSD by more than 40%. Meanwhile, it achieves comparable performance with the best baseline on the other one metric.
- **Time-invariant model performs poorly in simulating human activities.** Semi-Markov performs the worst in most cases. which indicates that the time-invariant assumption fails to describe behavior transition laws due to the existence of complex temporal patterns in daily activities.
- **Learning from raw data alone is insufficient for a realistic simulation.** The LSTM model has a poor performance on the metrics of *DailyAct* and *ActType*, which means errors

Table 3. Overall Performance of SAND and Baselines in Terms of the JSD-Based Metrics, and Lower Results Are Better

Dataset	Mobile Operator						Foursquare					
Metrics (JSD)	MacroInt	MicroInt	DailyAct	ActType	Weekday	Hour	MacroInt	MicroInt	DailyAct	ActType	Weekday	Hour
Semi-Markov	0.291	0.158	0.439	0.471	0.0042	0.051	0.334	0.055	0.485	0.101	0.0032	0.051
Hawkes	0.276	0.151	0.542	0.123	0.0039	0.051	0.073	0.024	0.530	0.026	0.0024	0.047
Neural Hawkes	0.026	0.143	0.125	**0.0063**	0.0036	0.052	0.072	0.041	0.119	0.012	0.0040	0.047
Neural JSDE	0.014	0.106	0.138	0.048	0.0033	0.051	0.041	0.033	**0.056**	0.0072	0.0022	0.046
THP	0.167	0.111	0.058	0.098	0.005	0.040	0.331	0.035	0.095	0.003	0.013	0.047
LSTM	0.110	0.136	0.513	0.342	0.0041	0.050	0.249	0.217	0.628	0.073	0.0033	0.051
SeqGAN	0.143	0.128	0.047	0.054	0.022	0.072	0.225	0.178	0.627	0.065	0.0034	0.051
GAIL	0.089	0.120	0.040	0.231	0.005	0.050	0.226	0.118	0.167	0.087	0.0049	0.062
SAND	**0.0096***	**0.084****	**0.025****	0.036	**0.002****	**0.009****	**0.018****	**0.014****	0.062	**0.0044****	**0.00032****	**0.0069****

Bold denotes the best results and underline denotes the second-best results. * and ** indicate that the improvement of our approach is significant for p-value<0.05 and p-value<0.01, respectively.

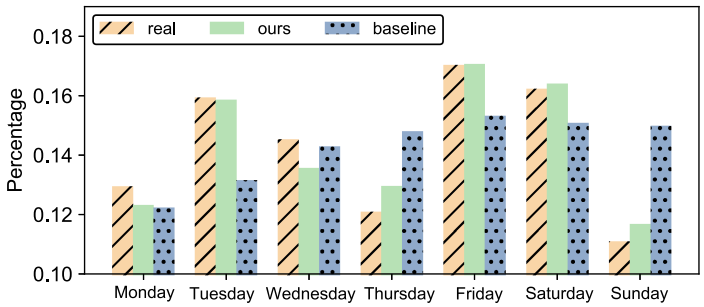


Fig. 9. Weekday distributions of the real data, generated data by SAND, and generated data by the best baseline.

can be accumulated in the step-by-step generation process. By contrast, SeqGAN and GAIL improve the performance by using reinforcement learning and adversarial learning. For the Foursquare dataset that is more sparse, their superiority is lost, which further suggests the instability of purely data-driven methods.

— **It is essential to model dynamic human needs.** The neural Hawkes, THP, and neural JSDE almost achieve the sub-optimal results on the two datasets, indicating the rationality of characterizing events in continuous time by temporal point processes. However, without investigating the deeper mechanism behind observed activities, their performance is still limited.

In summary, the performance on two activity datasets demonstrates the superiority of the SAND framework. By integrating the modeling of dynamic human needs in the GAIL framework, our approach achieves promising performance on activity simulation.

4.3.2 Distribution Visualization. Besides, to better illustrate the performance gain, we visualize the comparison distributions of the real data, generated data by SAND, and generated data by the best baseline, i.e., NJSDE. Here we select a representative temporal metric without loss of generality and show the results of the Foursquare dataset. Specifically, we take the weekday distribution as an example in Figure 9. As we can observe, the weekday distribution of our framework is very similar to the ground truth distribution.

4.3.3 Analysis of Model Efficiency. In this subsection, we compare the efficiency of different approaches, including training time, inference time, and model size. Considering that the performance of Semi-Markov and Hawkes models is far from satisfactory, we only compare the

Table 4. Numerical Comparison of Different Methods in Terms of Model Efficiency

Model	Training time	Inference time	Model size
NJSDE	≈ 7 hours	0.505 min	0.058 MB
SeqGAN	≈ 10 hours	0.050 min	0.044 MB
GAIL	≈ 12 hours	0.074 min	0.050 MB
SAND	≈ 8 hours	0.513 min	0.078 MB

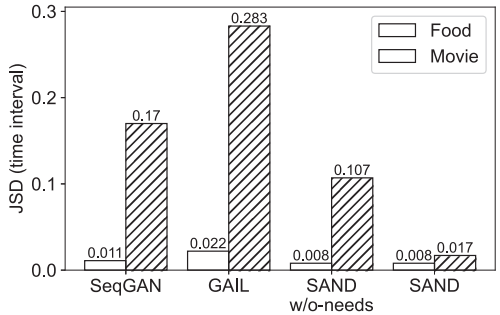


Fig. 10. Performance regarding the activity-wise JSD of the time interval.

model efficiency between our approach and state-of-the-art deep generative models, including NJSDE, SeqGAN, and GAIL. Note that the model size calculates all neural networks used in the framework, rather than the generator. Table 4 illustrates the numerical comparisons of our approach against other spatio-temporal techniques in terms of computation cost. As we can discover, NJSDE and our model, which are both conditioned on neural differential equations, have similar inference time and memory usage. These two methods have longer inference time due to the integral operation and Monte Carlo approximation when solving differential equations. Despite this, the inference time is acceptable, and they achieve remarkably better performance. The generators in SeqGAN and GAIL are RNN-based networks, so they have similar inference time and memory usage. Our method learns the need dynamics by neural differential equations as the policy learning in the Markov decision process. While taking advantage of continuous-time modeling and reinforcement learning, it does not add additional complexity obviously. The model converges normally in practice and can be effectively trained within 12 hours on a single GPU(RTX-2080Ti). We would like to note that our method achieves better performance with a comparable and acceptable overhead. Therefore, it can achieve a good balance between generation performance and model efficiency.

4.4 Performance towards Uneven Distributions

Technically, the incorporation of the knowledge-driven mechanism can overcome the drawbacks of existing purely data-driven methods such as being easily biased by uneven distribution. To demonstrate the ability of our method to tackle uneven distributions, we evaluate the performance regarding activity-wise JSD metrics. There exist uneven activity distributions in datasets (Section 4.1.1), so the performance variation of different activities is associated with the model's ability to deal with long-tail observational data. Specifically, we select two activities with the highest ratio and lowest ratio for the Foursquare dataset: Food and movie, then we compare the performance related to these two activities of purely data-driven solutions and our methods. We include SeqGAN, GAIL, and SAND w/o needs as data-driven methods. Figure 10 demonstrates the comparison

Table 5. Ablation Study on SAND Variants

Dataset	Mobile Operator						Foursquare					
Metrics (JSD)	MacroInt	MicroInt	DailyAct	ActType	Weekday	Hour	MacroInt	MicroInt	DailyAct	ActType	Weekday	Hour
SAND	**0.013***	**0.084****	**0.025****	**0.036***	**0.002****	**0.009****	**0.018***	**0.014****	0.062	**0.0044****	**0.00032****	**0.0069****
SAND - GAIL	0.013	0.116	0.085	0.040	0.0031	0.051	0.039	0.028	0.202	0.0051	0.0018	0.0092
SAND - need	0.014	0.116	0.085	0.039	0.0035	0.050	0.019	0.030	**0.0085**	0.0072	0.0021	0.048
SAND - pretrain	0.015	0.110	0.059	0.190	0.004	0.048	0.070	0.025	0.161	0.064	0.0020	0.044

Bold denotes the best results and underline denotes the second-best results. * and ** indicate that the improvement of our approach is significant for p-value<0.05 and p-value<0.01, respectively.

results. As we can observe, our method maintains similar performance for infrequent activities, while the first three data-driven methods suffer from significant degradation. The performance comparison demonstrates that our knowledge-driven modeling is capable of capturing unique patterns of different activities and is less biased by the overall but misleading activity patterns.

4.5 Ablation Studies (RQ2)

The proposed SAND framework consists of two key components: modeling need dynamics and solving the MDPs with GAIL. Besides, we also use the pre-training mechanism. To further validate whether they are indeed crucial for the final performance, we conduct ablation studies on two datasets by comparing the performance of three variants of SAND, including *SAND - need, SAND - GAIL, SAND - pretrain*. Specifically, *SAND - need* calculates the latent state as [26] without modeling hierarchical human needs, *SAND - GAIL* removes the GAIL training framework, and *SAND - pretrain* starts training from raw data without the pre-training mechanism.

The evaluation results are reported in Table 5. We can observe that SAND delivers the best performance on five metrics compared with the variants that are removed with specific designs. Without modeling need dynamics, the performance is reduced significantly, indicating the necessity to consider intrinsic motivation in human activity simulation. Besides, removing the GAIL framework also reduces the data fidelity, which suggests the strong modeling capabilities of generative adversarial mechanisms. In addition, the pre-training mechanism facilitates making full use of the activity data and enables our framework to preview the dependencies and regularities of daily activities before GAIL training, thus it also contributes to the final performance.

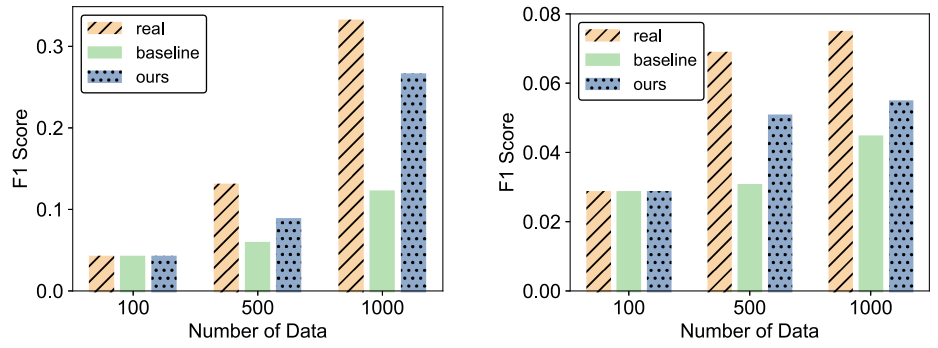
4.6 Case Studies (RQ3)

4.6.1 Practical Applications. In user-based applications, real-world activity records usually cannot be directly shared due to privacy issues. Under this circumstance, SAND can be used to generate synthetic data to mask sensitive information while retaining the usability of real data. To examine the utility of the generated synthetic data, we perform experiments with synthetic data of two categories:

— *Fully Synthetic Scenario.* Only synthetic data is used in applications, which provides more robust privacy protection.
— *Hybrid Scenario.* It combines real and synthetic data, which is widely used in data augmentation settings.

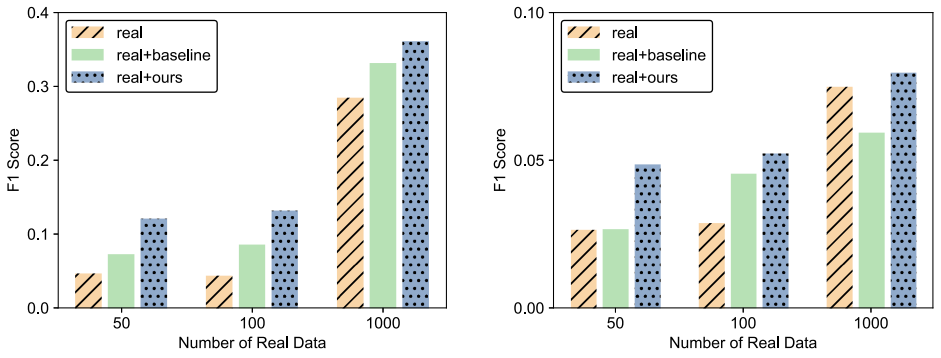
We select a representative application [24, 45] based on the activity data: activity prediction, which is fundamental to many activity-related problems, such as activity recommendation and planning.

We utilize a widely used model, LSTM with an attention mechanism, to predict individuals' future activity types based on their historical sequence. As shown in Figure 11, compared with the best baseline, the prediction performance on the dataset generated by our framework is much closer to the performance on the real data, showing the retained utility of the generated data.



(a) Mobile Operator Dataset. (b) Foursquare Dataset.

Fig. 11. Activity prediction in the fully synthetic scenario.

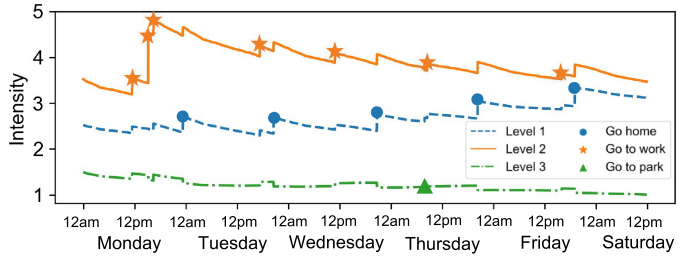


(a) Mobile Operator Dataset. (b) Foursquare Dataset.

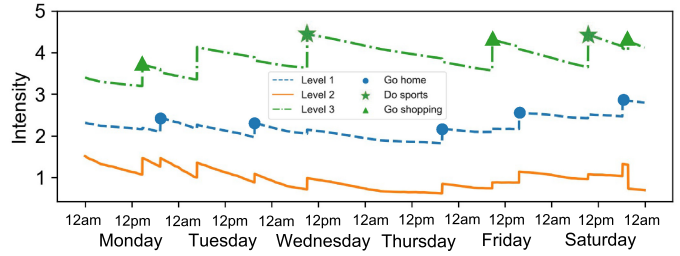
Fig. 12. Activity prediction in the hybrid scenario. For different number of real-world sequences, i.e., 50, 100, 1,000, we all add 1,000 generated sequences for data augmentation.

However, with only 100 sequences at our disposal, data generated using various approaches consistently exhibit subpar and similar performance. This implies this quantity is insufficient for effectively training a prediction model. Figure 12 illustrates that the model trained on the augmented data exhibits significantly better performance than that only trained on the real-world data. Meanwhile, the data augmented by SAND outperforms that by the best baseline. Moreover, the augmented data becomes more useful when the real-world data is of small scale, e.g., only with 50 or 100 real-world sequences. These results validate the practical value of the synthetic data.

4.6.2 Interpretability of Dynamic Needs. To validate whether SAND can provide insightful interpretability, we perform a case study on the learned intensity values of different need levels in the simulation process. Figure 13 illustrates the simulated activity sequences of two individuals for one week, together with the corresponding intensity values of three need levels. In terms of the model interpretability, we have two main observations. First, the proposed SAND can generate distinct but lifelike activity sequences that are hard to tell apart from real-world data. Specifically, comparing Figures 13(a) and 13(b), the two synthetic individuals lead quite personalized lifestyles. Individual 1 follows regular working routines with the intensity dynamics of the level-2 need



(a) Individual 1



(b) Individual 2

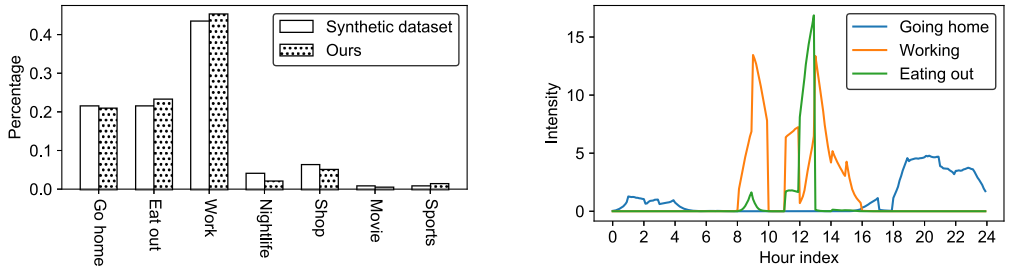
Fig. 13. Case study of two generated activity sequences and the learned intensity of different need levels. We select two representative individuals with different activity patterns.

varying periodically, while individual 2 enjoys more freedom without working, showing a constantly low intensity of the level-2 need. Second, SAND can simulate human daily activity in an interpretable way with need modeling. As observed from Figure 13, the occurrence of activity not only changes the intensity of the corresponding need level but also affects other levels, indicating that different need levels are interconnected by intensities derived from need states and trigger activities in a cooperative manner. In summary, the above observations demonstrate the interpretability of SAND for simulation outcomes, which is equally important in real-life applications.

4.7 Synthetic Evaluation

To better evaluate the proposed model's capability in modeling the latent needs, we conduct experiments on a synthetic dataset. The dataset is constructed manually, and individuals' daily life follows a working-recreation-going home rhythm. The occurrence time of specific activities exhibits a certain pattern, which is generated following rules: (1) The population consists of 80% white-collar workers and 20% students; (2) Between 6 am and 10 am, students commute to school; while white-collar workers head to their workplaces (second-level need); (3) The period from 11 am to 2 pm is designated for lunch (first-level need) for both groups; (4) Between 2 pm and 5 pm, they engage in either continued study or work; (5) After 5 pm, individuals decide whether to return home or engage in entertainment. These choices follow a Bernoulli distribution with probabilities of (0.6, 0.4) for workers and (0.8, 0.2) for students.

Figure 14(a) compares the distribution of different activities between the synthetic dataset and the dataset generated by our method. As we can observe, the two distributions are very similar to each other, thus, our model is able to capture the population-wise activity distribution. Besides, we explore the learned intensities of different activities to validate that the model can properly capture temporal patterns of different activities. Without loss of generality, we take three activities: going home, working, and eating out, as examples, which are shown in Figure 14(b). As we can observe,



(a) Comparison of Activity distributions.

(b) Intensities of example activities.

Fig. 14. Results of synthetic evaluation.

the intensity of working is high in the morning and afternoon, and the intensity of eating out peaks at noon. In the meantime, the intensity of going home increases late in the afternoon. These results are consistent with the logic of the synthetic dataset.

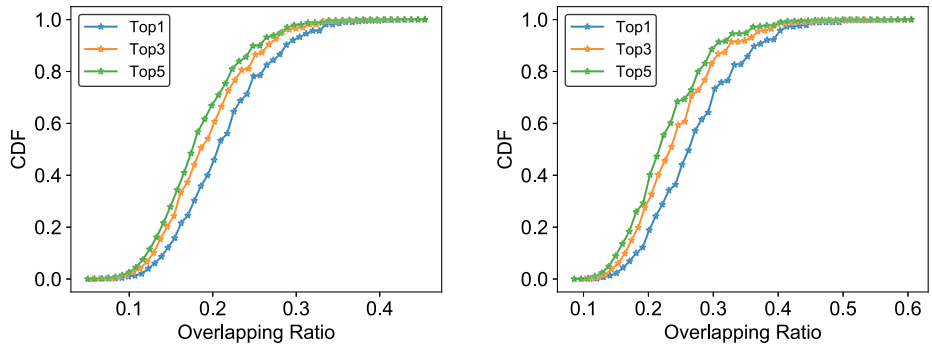
4.8 Protecting User Privacy (RQ4)

To prove that the generated daily activity sequence does not leak individual privacy, we perform experiments from comprehensive aspects as follows:

— *Uniqueness Testing* [13, 71]. It shows that the generated daily activity sequence is not a simple copy of the real sequence but a brand-new sequence.
— *Membership Inference Attacks* [40, 63]. Given a trained model and a set of data samples, the Membership inference attack infers whether those samples were included in the training dataset. Stronger privacy protection leads to a lower success rate.
— *Differential Privacy (DP)* [1, 3]. The released data should be similar whether a user's data is included in the training or not, which means the model are not highly dependent on any individual data.

4.8.1 Uniqueness Testing. We randomly select sequences from generated data and compare them with real sequences from the training set. We align the two sequences in the time dimension one by one and determine whether the activities at the corresponding time points are exactly the same. The overlapping ratio is defined as the ratio of the number of identical activities to the total sequence length. Next, we choose the real sequence that is the most similar to the generated one. We calculate the overlapping ration distribution and select the most similar 1, 3, and 5 real sequences for each generated sequence. Figure 15 presents the empirical cumulative distribution. For both datasets, more than 90% of the generated sequences cannot find any real sequences with more than 40% overlapping ratio, which demonstrates that our framework indeed learns to generate brand-new and unique trajectories rather than simply copying.

4.8.2 Membership Inference Attacks. The goal of such an attack is to determine whether those samples were in the training dataset. We follow the settings as in [40] and [63]. To control the impact of classification methods, we include three common-used classification algorithms: **Logistic Regression (LR)**, **Support Vector Machine (SVM)**, and **Random Forest (RF)** to perform attacks. The positive samples are individuals in the training data, while the negative samples are not. The feature is the overlapping ratios of multiple runs. The metric is the success rate, which is the percentage of successful trials in judging whether a sample is in the training dataset [40, 63]. Figure 16 shows the attack results. As we can observe, on both datasets, the attack success rate is less than 0.53, which means the attacker can hardly infer whether individuals are in the training



(a) Mobile Operator Dataset. (b) Foursquare Dataset.

Fig. 15. The results of Uniqueness testing.

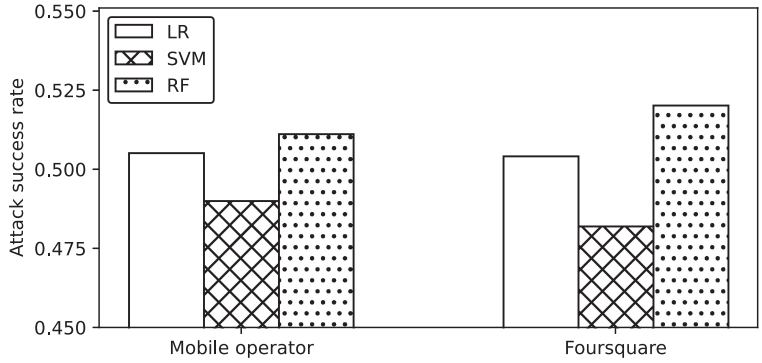


Fig. 16. Success rate of membership inference attack.

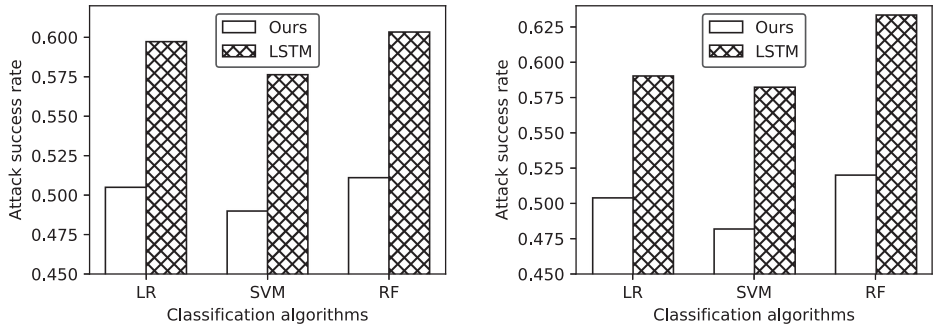
data from the generated data. The results above suggest the robustness of our framework to membership inference attacks.

In order to provide a more comprehensive evaluation of privacy preservation, we also examine the generated data from baseline methods. Without loss of generality, we present the results of membership inference attacks conducted on LSTM-generated data. Figure 17 compares the attack success rates across different classification algorithms for two types of generated datasets: those produced by our framework and those generated by LSTM models. As we can observe, the attack success rate regarding our approach is significantly lower than that for LSTM. This finding suggests that the adversarial training incorporated within the GAIL framework enhances the ability to obfuscate distinctions between real-world and synthetic data.

4.8.3 Differential Privacy. A model M is (ϵ, δ) differentially private if for any pair of datasets D and D' that differ in the activity data of a single user, it holds that [1, 3]:

$$M(z; D) \leq e^{\epsilon} M(z; D') + \delta$$

For the output z, $M(z, D)$ denotes the probability distribution of z with the data D as the input. Smaller values of ϵ and δ give more privacy. We examine the privacy budget of our proposed model from the perspective of changes in the overlapping ratio. Specifically, the overlapping ratio of each individual under the conditions that this individual is contained in the training dataset or



(a) Mobile Operator Dataset. (b) Foursquare Dataset.

Fig. 17. The results of Uniqueness testing.

Table 6. Statistics of ϵ for the Two Datasets

ϵ	Mean	Median	90th percentile
Mobile operator	1.429	0.872	3.747
Foursquare	0.9735	0.504	2.754

not is modeled by two Gaussian distributions respectively, which are then regarded as $M(z, D)$ and $M(z, D')$ to calculate the privacy budget ϵ. For each user, we calculate the ϵ using TensorFlow Privacy [1, 3]. We list the statistical results of ϵ in Table 6. We can observe that without any additional privacy-preserving mechanism, our model is able to have an average privacy budget $\epsilon \approx 1$, which is typically considered a reasonable operating point for generative models [40], e.g., Apple uses a privacy budget of 4.0.[6] The privacy budget can be further enhanced by introducing DP-SGD [1] or DP-GAN [40, 70].

5 RELATED WORK

5.1 Human Activity Simulation

Solutions for activity simulation are mainly agent-based modeling [42] with rule-based methods [30–32, 47, 48, 53]. Specifically, these methods assume that human activities can be described by limited parameters with explicit physical meaning and are governed by transition rules based on psychology and social science theories. With simplified assumptions of human behaviors, agents in the system can be assigned different goals, then they take actions to maximize different attributes. For example, Kim et al. [32] propose that human actions are triggered by a cause and give rise to corresponding effects. Besides, considering the multiple behaviors, the priorities of behaviors are determined based on Maslow's hierarchy of needs [30–32]. Despite the promising performance under some circumstances, rule-based methods fail to capture complicated activity patterns due to relying on simplified assumptions and thus usually fail to simulate activities in reality. The purpose of activity simulation is different from that of activity prediction [24, 45, 73]. The former emphasizes the simulation results to reproduce and reflect characteristics of real data, but should not be too similar to real data with the goal of protecting user privacy, while the latter highlights to what extent the model can recover the real data. Although deep learning approaches are proposed for activity prediction [25, 49], the problem of simulating daily activities has been barely explored.

[6] https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf

Moreover, it's important to distinguish between activity simulation and itinerary or trip planning processes [9, 27]. Activity simulation involves the generation of realistic activity sequences for a large population, whereas itinerary and trip planning revolve around creating trip plans that adhere to user-provided constraints. These constraints typically include a selection of points of interest (POIs), as well as considerations such as time and budget. In parallel, trip recommendation [28] and path prediction [81] focus on predicting the next location based on historical POI data. In contrast, activity simulation stands apart by directly generating entire activity sequences without relying on prior historical activities. Our problem is also different from time series prediction [80], as we generate activities at irregular time intervals. This distinction underscores the unique objectives and methodologies employed in these various aspects of location-based recommendation systems.

5.2 Deep Generative Models for Activity Simulation

Deep generative models, such as **generative adversarial networks (GAN)** [16] and **variational autoencoder (VAE)** [33], are promising solutions to simulation. Previous studies [22, 51, 62, 68, 79] have also explored the ability of GAIL to simulate human decision process. Besides, a series of neural temporal point process models [10, 44, 76, 82] are proposed to model discrete events. Although these models are mainly for discrete event prediction, the learned probability distribution provides opportunities to perform event generation by the sampling operation. Recently, Gupta et al. [19] propose attention-based temporal point process flows to model goal-directed activity sequences. However, it is not appropriate for our research problems as daily activities cannot be represented as a sequence of actions performed to achieve an explicit goal. We propose a knowledge-driven framework based on GAIL, and the incorporation of psychological knowledge is realized by leveraging an ODE-based temporal point process.

5.3 Generative Models in Machine Learning

Generative models can be categorized into two types: explicitly generative models and implicit generative models. For explicitly generative models, the idea is to minimize some notion of divergence in terms of the data distribution [18]. For example, minimizing the KL divergence between the generated data by the model and the observational data is equivalent to performing **maximum likelihood estimation (MLE)** on the observational data [46]. In contrast, generative models based on adversarial learning have recently attracted much attention, such as GAN, GAIL, and SeqGAN [16, 20, 74]. The objective is to generate data that cannot be distinguished from the real data, thus, these models no longer need to specify an explicit density for the observational data. Point process models [23, 44] are generally implicit generative models where the density of the event is estimated. GAIL [20] is a typically implicitly generative model via training a policy function and a reward function adversarially. Our proposed framework models the need dynamics by characterizing the activity sequence as a point process and utilizes differential equations to describe the temporal evolution. The pre-training on the point process leverages maximum likelihood, and the adversarial learning in GAIL further encourages the generation of more realistic samples. The key idea of modeling dynamics human needs allows for hybrid learning by using maximum likelihood and adversarial training to improve the final performance.

6 DISCUSSION AND CONCLUSION

In this article, we investigate the individual activity simulation problem by proposing a novel framework SAND, which integrates deep generative models with well-respected psychological theories. Extensive experiments on two real-world datasets show the superior performance of the proposed framework. Our framework is not strictly limited to Maslow's theories, instead, what we

highlight is leveraging neural networks to learn the driving force behind human daily activities, and the choice of knowledge or theory related to such driving force is quite flexible. For example, ERF theory [2] claims that there exist three levels behind activities, including existence, relatedness, and growth; some other theories [59] propose that several specific travel purposes lead to daily activities. Importantly, effective modeling of human needs makes it possible to understand human behaviors at a deeper level, which not only benefits the activity simulation in this work but also contributes to many other problems of psychology-informed user modeling.

The proposed framework brings significant advantages to the state-of-the-art spatio-temporal graph learning frameworks [36, 41, 66, 78]. The activity generator provides invaluable insights into dynamic spatio-temporal graph learning within continuous domains, enhancing the comprehension of intricate, long-range spatio-temporal dependencies. Furthermore, the generated activity data holds the potential to revolutionize decision-making across a multitude of domains. In the realm of traffic prediction, the simulation of realistic movement patterns, including vehicle flow and pedestrian activities, empowers practitioners to assess model robustness and strategy effectiveness under diverse conditions. As a result, decision-makers can make more informed choices related to traffic management, infrastructure enhancements, and route optimization. In urban planning, synthetic human activity data proves instrumental in both the design and evaluation of urban spaces. In the domain of social network analysis, synthetic human activity data equips researchers to delve into the dynamics of social interactions, information dissemination, and the formation of communities within networks.

For future work, it is promising to incorporate more advanced generative models to simulate daily activities, such as diffusion models [75] and large language models [52].

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *CCS*. 308–318.

[2] Clayton P. Alderfer. 1969. An empirical test of a new theory of human needs. *Organizational Behavior and Human Performance* 4, 2 (1969), 142–175.

[3] Galen Andrew, Steve Chien, and Nicolas Papernot. 2019. TensorFlow privacy. https://blog.tensorflow.org/2019/03/introducing-tensorflow-privacy-learning.html

[4] Theo Arentze, Frank Hofman, Henk van Mourik, and Harry Timmermans. 2000. ALBATROSS: Multiagent, rule-based model of activity pattern decisions. *Transportation Research Record* 1706, 1 (2000), 136–144.

[5] Joshua Auld and Abolfazl Kouros Mohammadian. 2012. Activity planning processes in the agent-based dynamic activity planning and travel scheduling (ADAPTS) model. *Transp. Res. Part A Policy Pract.* 46, 8 (2012), 1386–1403.

[6] John J. Bartko. 1966. The intraclass correlation coefficient as a measure of reliability. *Psychological Reports* 19, 1 (1966), 3–11.

[7] John L. Bowman and Moshe E. Ben-Akiva. 2001. Activity-based disaggregate travel demand model system with activity schedules. *Transportation Research Part A: Policy and Practice* 35, 1 (2001), 1–28.

[8] Wei-Lun Chang and Soe-Tsyr Yuan. 2008. A synthesized model of Markov chain and ERG theory for behavior forecast in collaborative prototyping. *JITTA* 9, 2 (2008), 5.

[9] Gang Chen, Sai Wu, Jingbo Zhou, and Anthony K. H. Tung. 2013. Automatic itinerary planning for traveling services. *IEEE Transactions on Knowledge and Data Engineering* 26, 3 (2013), 514–527.

[10] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366* (2018).

[11] Kae H. Chung. 1969. A Markov chain model of human needs: An extension of Maslow's need theory. *Academy of Management Journal* 12, 2 (1969), 223–234.

[12] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. 2019. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. *arXiv preprint arXiv:1905.12374* (2019).

[13] Yves-Alexandre De Montjoye, César A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports* 3, 1 (2013), 1–5.

[14] Dick Ettema, Aloys Borgers, and Harry Timmermans. 1993. Simulation model of activity scheduling behavior. *Transportation Research Record* (1993), 1–1.

[15] Jie Feng, Zeyu Yang, Fengli Xu, Haisu Yu, Mudan Wang, and Yong Li. 2020. Learning to simulate human mobility. In *KDD*. 3426–3433.
[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *NIPS* 27 (2014).
[17] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).
[18] Aditya Grover, Manik Dhar, and Stefano Ermon. 2018. Flow-GAN: Combining maximum likelihood and adversarial learning in generative models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[19] Vinayak Gupta and Srikanta Bedathur. 2022. ProActive: Self-attentive temporal point process flows for activity sequences. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 496–504.
[20] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems* 29 (2016), 4565–4573.
[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[22] Yuchen Wu Depeng Jin Lina Yao Huandong Wang, Changzheng Gao and Yong Li. 2023. PateGail: A privacy-preserving mobility trajectory generator with imitation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
[23] Martin Jacobsen and Joseph Gani. 2006. *Point Process Theory and Applications: Marked Point and Piecewise Deterministic Processes*. Springer.
[24] Vikramaditya Jakkula and Diane J. Cook. 2007. Mining sensor data in smart environment for temporal activity prediction. *KDD* (2007).
[25] Neziha Jaouedi, Francisco J. Perales, José Maria Buades, Noureddine Boujnah, and Med Salim Bouhlel. 2020. Prediction of human activities based on a new structure of skeleton features and deep learning model. *Sensors* 20, 17 (2020), 4944.
[26] Junteng Jia and Austin R. Benson. 2019. Neural jump stochastic differential equations. *arXiv preprint arXiv:1905.10403* (2019).
[27] Linlang Jiang, Jingbo Zhou, Tong Xu, Yanyan Li, Hao Chen, and Dejing Dou. 2022. Time-aware neural trip planning reinforced by human mobility. In *Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
[28] Linlang Jiang, Jingbo Zhou, Tong Xu, Yanyan Li, Hao Chen, Jizhou Huang, and Hui Xiong. 2021. Adversarial neural trip recommendation. *arXiv preprint arXiv:2109.11731* (2021).
[29] Sidney Katz. 1983. Assessing self-maintenance: Activities of daily living, mobility, and instrumental activities of daily living. *Journal of the American Geriatrics Society* 31, 12 (1983), 721–727.
[30] Hamdi Kavak, Joon-Seok Kim, Andrew Crooks, Dieter Pfoser, Carola Wenk, and Andreas Züfle. 2019. Location-based social simulation. In *SSTD*. 218–221.
[31] Joon-Seok Kim, Hyunjee Jin, Hamdi Kavak, Ovi Chris Rouly, Andrew Crooks, Dieter Pfoser, Carola Wenk, and Andreas Züfle. 2020. Location-based social network data generation based on patterns of life. In *MDM*. IEEE, 158–167.
[32] Joon-Seok Kim, Hamdi Kavak, Umar Manzoor, Andrew Crooks, Dieter Pfoser, Carola Wenk, and Andreas Züfle. 2019. Simulating urban patterns of life: A geo-social data generation framework. In *SIGSPATIAL*. 576–579.
[33] Diederik P. Kingma and Max Welling. 2019. An introduction to variational autoencoders. *Found. Trends Mach. Learn.* 12, 4 (2019), 307–392.
[34] Ryuichi Kitamura, Eric I. Pas, Clarisse V. Lula, T. Keith Lawton, and Paul E. Benson. 1996. The sequenced activity mobility simulator (SAMS): An integrated approach to modeling transportation, land use and air quality. *Transportation* 23, 3 (1996), 267–291.
[35] Patrick J. Laub, Thomas Taimre, and Philip K. Pollett. 2015. Hawkes processes. *arXiv preprint arXiv:1507.02822* (2015).
[36] Lincan Li, Kaixiang Yang, Fengji Luo, and Jichao Bi. 2023. STS-CCL: Spatial-temporal synchronous contextual contrastive learning for urban traffic forecasting. *arXiv preprint arXiv:2307.02507* (2023).
[37] Yuxuan Liang, Kun Ouyang, Hanshu Yan, Yiwei Wang, Zekun Tong, and Roger Zimmermann. 2021. Modeling trajectories with neural ordinary differential equations. In *IJCAI*. 1498–1504.
[38] Nikolaos Limnios and Gheorghe Oprisan. 2012. *Semi-Markov Processes and Reliability*. Springer.
[39] Jianhua Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory* 37, 1 (1991), 145–151.
[40] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2020. Using GANs for sharing networked time series data: Challenges, initial promise, and open questions. In *IMC*. 464–483.
[41] Yihong Ma, Patrick Gerard, Yijun Tian, Zhichun Guo, and Nitesh V. Chawla. 2022. Hierarchical spatio-temporal graph neural networks for pandemic forecasting. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1481–1490.
[42] Charles M. Macal and Michael J. North. 2005. Tutorial on agent-based modeling and simulation. In *WSC*. IEEE, 14–pp.
[43] Abraham Harold Maslow. 1943. A theory of human motivation. *Psychological Review* 50, 4 (1943), 370.
[44] Hongyuan Mei and Jason Eisner. 2016. The neural Hawkes process: A neurally self-modulating multivariate point process. *arXiv preprint arXiv:1612.09328* (2016).

[45] Bryan Minor, Janardhan Rao Doppa, and Diane J. Cook. 2015. Data-driven activity prediction: Algorithms, evaluation methodology, and applications. In *KDD*. 805–814.
[46] Shakir Mohamed and Balaji Lakshminarayanan. 2016. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483* (2016).
[47] Goran Murić, Alexey Tregubov, Jim Blythe, Andrés Abeliuk, Divya Choudhary, Kristina Lerman, and Emilio Ferrara. 2020. Massive cross-platform simulations of online social networks. In *AAMAS*. 895–903.
[48] Michael J. North, Charles M. Macal, James St. Aubin, Prakash Thimmapuram, Mark Bragen, June Hahn, James Karr, Nancy Brigham, Mark E. Lacy, and Delaine Hampton. 2010. Multiscale agent-based consumer market modeling. *Complexity* 15, 5 (2010), 37–47.
[49] Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-Garadi, and Uzoma Rita Alo. 2018. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications* 105 (2018), 233–261.
[50] Kun Ouyang, Reza Shokri, David S. Rosenblum, and Wenzhuo Yang. 2018. A non-parametric generative model for human trajectories. In *IJCAI*. 3812–3817.
[51] Menghai Pan, Weixiao Huang, Yanhua Li, Xun Zhou, and Jun Luo. 2020. xGAIL: Explainable generative adversarial imitation learning for explainable human decision analysis. In *KDD*. 1334–1343.
[52] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442* (2023).
[53] Andreas Züfle, Hamdi Kavak, Joon-Seok Kim, Dieter Pfoser, Carola Wenk, Umar Manzoor, and Andrew Crooks. 2021. *Towards Large-Scale Agent-Based Geospatial Simulation.*
[54] Jan-Hendrik Prinz, Hao Wu, Marco Sarich, Bettina Keller, Martin Senne, Martin Held, John D. Chodera, Christof Schütte, and Frank Noé. 2011. Markov models of molecular kinetics: Generation and validation. *The Journal of Chemical Physics* 134, 17 (2011), 174105.
[55] Lawrence Rabiner and Biinghwang Juang. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine* 3, 1 (1986), 4–16.
[56] John Rauschenberger, Neal Schmitt, and John E. Hunter. 1980. A test of the need hierarchy concept by a Markov model of change in need strength. *Administrative Science Quarterly* (1980), 654–670.
[57] W. W. Recker and G. S. Root. 1981. Toward a dynamic model of individual activity pattern formulation. (1981).
[58] Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. 2019. Latent ODEs for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907* (2019).
[59] Ilan Salomon. 1985. Telecommunications and travel: Substitution or modified mobility? *Journal of Transport Economics and Policy* (1985), 219–235.
[60] Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. 2019. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790* (2019).
[61] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[62] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. 2019. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *AAAI*, Vol. 33. 4902–4909.
[63] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.
[64] Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *ACL*. 1501–1510.
[65] Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning*. Vol. 135.
[66] Xuxiang Ta, Zihan Liu, Xiao Hu, Le Yu, Leilei Sun, and Bowen Du. 2022. Adaptive spatio-temporal graph neural network for traffic forecasting. *Knowledge-Based Systems* 242 (2022), 108199.
[67] Chen-Ya Wang, Yueh-Hsun Wu, and Seng-Cho T. Chou. 2010. Toward a ubiquitous personalized daily-life activity recommendation service with contextual information: A services science perspective. *Inf. Syst. E-Bus. Manag.* 8, 1 (2010), 13–32.
[68] Hua Wei, Dongkuan Xu, Junjie Liang, and Zhenhui Li. 2021. How do we move: Modeling human movement with system dynamics. In *AAAI*.
[69] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. 2017. Modeling trajectories with recurrent neural networks. In *IJCAI*.
[70] Chugui Xu, Ju Ren, Deyu Zhang, Yaoxue Zhang, Zhan Qin, and Kui Ren. 2019. GANobfuscator: Mitigating information leakage under GAN via differential privacy. *IEEE TIFS* 14, 9 (2019), 2358–2371.
[71] Fengli Xu, Zhen Tu, Yong Li, Pengyu Zhang, Xiaoming Fu, and Depeng Jin. 2017. Trajectory recovery from ash: User privacy is not preserved in aggregated mobility data. In *WWW*. 1241–1250.

[72] Dingqi Yang, Daqing Zhang, Vincent W. Zheng, and Zhiyong Yu. 2014. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. *TSMC* 45, 1 (2014), 129–142.
[73] Jihang Ye, Zhe Zhu, and Hong Cheng. 2013. What's your next move: User activity prediction in location-based social networks. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 171–179.
[74] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, Vol. 31.
[75] Yuan Yuan, Jingtao Ding, Chenyang Shao, Depeng Jin, and Yong Li. 2023. Spatio-temporal diffusion point processes. *arXiv preprint arXiv:2305.12403* (2023).
[76] Yuan Yuan, Jingtao Ding, Huandong Wang, Depeng Jin, and Yong Li. 2022. Activity trajectory generation via modeling spatiotemporal dynamics. In *KDD*. 4752–4762.
[77] Yuan Yuan, Huandong Wang, Jingtao Ding, Depeng Jin, and Yong Li. 2023. Learning to simulate daily activities via modeling dynamic human needs. In *Proceedings of the ACM Web Conference 2023*. 906–916.
[78] Qianru Zhang, Chao Huang, Lianghao Xia, Zheng Wang, Zhonghang Li, and Siuming Yiu. 2023. Automated spatio-temporal graph contrastive learning. In *Proceedings of the ACM Web Conference 2023*. 295–305.
[79] Xin Zhang, Yanhua Li, Xun Zhou, and Jun Luo. 2019. Unveiling taxi drivers' strategies via cgail: Conditional generative adversarial imitation learning. In *ICDM*. IEEE, 1480–1485.
[80] Jingbo Zhou and Anthony K. H. Tung. 2015. Smiler: A semi-lazy time series prediction system for sensors. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 1871–1886.
[81] Jingbo Zhou, Anthony K. H. Tung, Wei Wu, and Wee Siong Ng. 2013. A "semi-lazy" approach to probabilistic path prediction in dynamic environments. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 748–756.
[82] Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. 2020. Transformer hawkes process. In *ICML*. PMLR, 11692–11702.

Received 14 June 2023; revised 23 October 2023; accepted 20 November 2023