User Device Interaction Prediction via Relational Gated Graph Attention Network and Intent-aware Encoder

Jingyu Xiao*
Tsinghua Shenzhen International Graduate School
Peng Cheng Laboratory
Shenzhen, China
jy-xiao21@mails.tsinghua.edu.cn

Qingsong Zou*
Tsinghua Shenzhen International Graduate School
Peng Cheng Laboratory
Shenzhen, China
zouqs21@mails.tsinghua.edu.cn

Qing Li†
Peng Cheng Laboratory
Shenzhen, China
liq@pcl.ac.cn

Dan Zhao
Peng Cheng Laboratory
Shenzhen, China
zhaod01@pcl.ac.cn

Kang Li
Jilin University
Changchun, China
likang9920@mails.jlu.edu.cn

Wenxin Tang
Jilin University
Changchun, China
tangwx9919@mails.jlu.edu.cn

Runjie Zhou
Shandong University
Jinan, China
runjiezhou@mail.sdu.edu.cn

Yong Jiang
Tsinghua Shenzhen International Graduate School
Peng Cheng Laboratory
Shenzhen, China
jiangy@sz.tsinghua.edu.cn

ABSTRACT

With the booming of smart home market, intelligent Internet of Things (IoT) devices have been increasingly more involved in home life. To improve the user experience of smart home, some prior works have explored how to use time series analysis technology for predicting the interaction between users and devices. However, existing solutions have inferior User Device Interaction (UDI) prediction accuracy, as they fail to consider the complex heterogeneous device transitions, multiple intents of a user and multi-level periodicity of user behaviors. In this paper, we present DeepUDI, a novel approach for accurate UDI prediction. First, we propose Relational Gated Graph Attention Network (RGGAT) to learn embedding of device and device control while considering complex heterogeneous temporal transitions. Second, we propose Intent-aware Encoder (IAE) to encode multiple intents of users via capsule networks. Third, we design a Historical Attention Mechanism (HAM) to capture the multi-level periodicity by aggregating the current sequence and the historical sequence representations through the attention mechanism. Comprehensive experiments on four real-world datasets show that DeepUDI consistently outperforms state-of-the-art baselines and also offers highly interpretable results.

KEYWORDS

User Device Interaction;Graph Neural Networks;Capsule Networks

*The first two authors have equal contribution.
†Qing Li is the corresponding author.

ACM Reference Format:
Jingyu Xiao, Qingsong Zou, Qing Li, Dan Zhao, Kang Li, Wenxin Tang, Runjie Zhou, and Yong Jiang. 2023. User Device Interaction Prediction via Relational Gated Graph Attention Network and Intent-aware Encoder. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

With fast-evolving IoT solutions, the number of smart devices in homes have soared, expected to reach 5 billion by 2025 [13]. The emergence of cloud platforms also allows IoT sensors and actuators to better assist users in various home living activities. In this process, users' use of devices, i.e., device controls, can reflect their behavioral habits and intents. Exploiting the relationship between user behavior habits, intents and their use of devices bring about opportunities from various perspectives. For service providers, such as vendors, knowing users' living habits through their device usage histories can offer insights for improving user experience. From the perspective of device intelligence, prediction of user behavior can help intelligent platforms recommend actions that users may like to perform, or recommend automation rules that users may be interested in, such as *lock the back door when the front door is closed outside*. From the perspective of user behavior analysis, accurate user behavior prediction can be used for abnormal user behavior identification, elderly/disabled care.

Motivated by these, some prior studies have analyzed user behaviors at home and adopted user behavior prediction technologies for different purposes. [11, 20] infer the users' behavioral intents from their sequential operations on the device, and generate alternative automation rules and next behavior recommendations. [1, 15, 27] analyze the behavior patterns of users at home, and observe the status of connected smart home entities (sensors and devices) through different user activities and usage patterns for the benefit of elderly care and abnormal behavior recognition. However, there are three main challenges in the prediction of User Device Interaction (UDI) that have not been properly addressed in smart home scenarios.

First, complex heterogeneous transitions between devices user accessed makes it difficult for models to learn behavior representations. On the one hand, there are transitions not only between consecutive devices, but also in broader contexts (i.e., other devices in the behavior sequence). For example, as shown in Figure 1, the user interacts with the water valve at time t_1 and t_6. All devices between time t_1 and t_6, rather than only the devices immediately next to t_1 and t_6, have transitions with the water valve. On the other hand, the transitions among different devices are heterogeneous, that is, caused by different device controls. For example, the transition from the oven to the microwave can be caused by the device control "turn on the microwave", or caused by the device control "turning off the microwave" (e.g., the user's reaction upon receiving a finish notification from the microwave).

Second, there are often multiple intents in a UDI sequence. Relying purely on a user's sequential behavior without considering intents may lead to wrong predictions. For example, a user may cook while doing laundry because of the long wash cycle of the washing machine, as shown in Figure 1. There are two *intents*, i.e., laundry and cooking, in the behavior sequence. Without considering user's multiple intents, after observing the behavior from t_1 to t_4, the next behavior is likely to be predicted as "start dish washer", because recent behaviors "switch on oven" and "switch on microwave" are both cooking-related. However, in fact, before the meal is ready, the washing machine cycle has finished, and the user's next behavior should be "turn off washing machine".

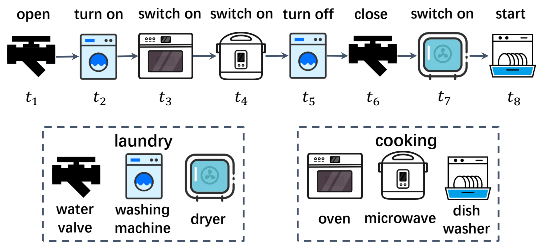


Figure 1: An example of user sequential behavior with two intents: laundry (watervalve/washing machine/dryer) and cooking (oven/microwave/dish washing machine).

Third, the multi-level periodicity in user behavior makes it difficult for models to predict the interactions. For example, the bedtime of a user, which determines when sleep-related interactions ("close curtain"/"turn off light") occur, is not fixed. As shown in Figure 2, a user may leave work on time on Wednesday and Thursday, and go to bed early but may work overtime once every 3 days, e.g., on Tuesday and Friday, and therefore go to bed later. Every Saturday night, the user may want to stay up late for games, and sleep even later. In this example, the user's behavior is of different periodicity, that is, working overtime every three days, and playing games till late at night every Saturday, causing fluctuations in sleeping time over the week.

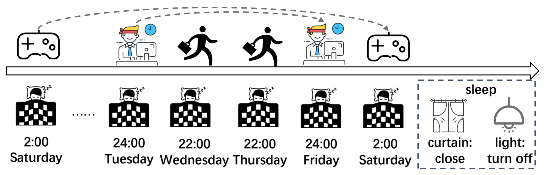


Figure 2: The multi-level periodicity in the user behavior.

To address the above challenges, in this paper, we propose DeepUDI, a novel approach for accurate UDI prediction. The design of DeepUDI mainly includes the following three ideas. First, DeepUDI designs a multi-modal embedding layer to encode device control along with device and time to fully account for contextual information. Specifically, we propose a Relational Gated Graph Attention Network (RGGAT) to learn the representation of devices and device controls to mine the complex heterogeneous transitions among different devices. To capture periodic patterns of time, we leverage Time2Vec [8] to learn the temporal representations. Second, we propose an Intent-aware Encoder (IAE) to encode multiple intents of users from the UDI sequence. DeepUDI views behaviors of different intents as different primary capsules, and learns multi-intent representations of users through the capsule networks [19], then an inter-intent aggregation mechanism is applied to learn weights of different intents for aggregating representations. Third, we design a Historical Attention Mechanism (HAM) to capture the multi-level periodicity by aggregating the current sequence and the historical sequence representations through the attention mechanism. Our main contributions are summarized as follows:

- We propose **RGGAT**, an architecture based on graph attention networks, which converts user behavior sequences into relational sequence graphs to learn complex heterogeneous transitions among devices.
- We propose **IAE**, an encoder based on capsule networks, which view behaviors as primary capsules and learn multi-intent representations of users by dynamic routing and inter-intent aggregation.
- We propose **HAM** to capture multi-level periodicity of user behaviors by applying attention mechanism between current and historical user behavior sequences.

2 RELATED WORKS

Given a behavior sequence, predicting the next behavior of the user is called the sequential prediction problem (e.g., recommendation [12] and human mobility prediction [4]). Traditional methods such as Markov chains (MC) [5], Matrix Factorization (MF) [14] and FPMC [18] are applied in sequential prediction in earlier years. [5] builds a transition matrix between states to make predictions. [14] factorizes the users-locations matrix to generate user general preferences to complete human next location prediction. [18] combines first-order Markov Chains and Matrix Factorization to model both sequential behaviors and general interests of users for the sequential recommendation. With the development of deep learning, deep neural networks such as Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Graph Neural Networks (GNN) and Transformers [23] have been adopted in sequential user behavior prediction. CA-RNN [12] and SIAR [17] incorporate contextual information into the RNN for the sequential recommendation. Caser [21] employs CNN in both time-axis and feature-axis to capture temporal dynamics in a sequential recommendation. [25, 26] apply gated GNN to capture complex transition patterns among nodes for achieving the session-based recommendation. SASRec [7] utilizes unidirectional transformers to capture sequential patterns in sequences while considering the importance of correlations between behaviors. SmartSense [6] utilizes a two-stage encoder for IoT action recommendations. However, these methods either do not consider the complex transitions between behaviors, or do not consider the multi-intent and multi-periodicity of user behaviors.

3 DEEPUDI OVERVIEW

Let $\mathcal{D}$ denote a set of devices, C denote a set of device controls, $\mathcal{I}$ denote a set of intents and $\mathcal{S}$ denote a set of behavior sequences.

Definition 1. *(Behavior) The behavior $b = (t, d, c, i)$, is a 4-tuple consisting of time t, device $d \in \mathcal{D}$, device control $c \in C$, and intent $i \in \mathcal{I}$. For example, the behavior b = (2022-10-15 11:30, oven, oven:switch on, cooking) describes the behavior "turn on the oven" at 11:30 on 2022-10-15, with the intent of cooking.*

Definition 2. *(Behavior Sequence) The behavior sequence $s = [b_1, b_2, \cdots, b_n] \in \mathcal{S}$ is a list of behaviors. b is ordered by timestamps, and n is the length of s.*

We describe the User Device Interaction (UDI) prediction problem definition as follows.

Problem 1. *(UDI Prediction) Given a previous sequence $s \in \mathcal{S}$, predict the next behavior b_{n+1} in the behavior sequence.*

In this paper, we divide the behavior into fixed time intervals. Since the device control contains both device and intent information (e.g., "oven:switch on" indicates that the device is oven and the user's intent is cooking), the problem is simplified to predict the next device control c_{n+1} in the next time interval.

To address the three challenges mentioned above, we propose DeepUDI, depicted in Figure 3. DeepUDI mainly consists of a multi-modal embedding layer (§4), an intent-aware encoder layer (§5) and a historical attention layer (§6).

First, the behavior sequence data is fed into a **multi-modal embedding layer** to extract behavior representations, where device and device control embedding are learned by Relational Gated Graph Attention Network (RGGAT) and time embedding are learned by Time2Vec [8]. Then, the behavior representations are input to the **intent-aware encoder layer** to extract multi-intent representations of users for the sequence representations. After getting sequence representations, a **historical attention layer** is applied to extract history sequence representations, which are connected with the current sequence representation for the final prediction.

4 MULTI-MODAL EMBEDDING LAYER

We design a multi-modal embedding layer to encode device control with device and time to fully account for contextual information.

4.1 Time Embedding

Due to the continuity of timestamps, it is impractical to learn temporal representations directly from timestamps. We express time as hour of the day and day of the week, as they are shown to affect users' device controls [6]. We leverage Time2Vec [8] to learn time embedding because it can capture both periodic and non-periodic patterns and is invariant to time rescaling. For a given scalar notion of time τ, the embedding $\mathbf{t2v}(\tau)$ of size L can be defined as follows:

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i \tau + \varphi_i, & \text{if } i = 0 \\ \mathcal{F}\left(\omega_i \tau + \varphi_i\right), & \text{if } 1 \leq i \leq L-1 \end{cases} \tag{1}$$

where $\mathbf{t2v}(\tau)[i]$ denotes the i-th element of $\mathbf{t2v}(\tau)$, $\mathcal{F}$ is a periodic activation function, i.e., a sine function, ω_i and φ_i are learnable parameters. For instance, a sine function $sin(\omega\tau + \phi)$ with $\omega = \frac{2\pi}{7}$ repeats every 7 days (assuming τ indicates days) and can potentially model weekly patterns. Let $\mathbf{e}_{dw}, \mathbf{e}_h \in \mathbb{R}^L$ denote the embeddings of day of the week and hour of the day which are obtained by Time2Vec, respectively.

4.2 Device and Device Control Embeddings

[25] proves that Gated Graph Neural Networks (GGNN) can mine the transitions between different nodes in a sequence graph. However, GGNN cannot be directly applied to our scenario because the transitions between different devices are heterogeneous (different device controls). Inspired by [2], which incorporates relational information into Graph Attention Networks (GAT), we propose **Relational Gated Graph Attention Network (RGGAT)** to learn device and device control embeddings from the relational sequence graphs constructed by behavior sequences.

4.2.1 Relational Sequence Graph (RS-Graph) Construction. A behavior sequence s can be modeled as a relational directed graph $G_s(V_s, E_s)$ with $R = |\mathscr{R}|$ relation types and N nodes. Each node in the graph represents a device, each edge $(d_{n-1}, d_n) \in E_s$ indicates that the user accesses device d_n after accessing device d_{n-1} and each device control represents a relation $r \in C$. Specifically, let $\mathbf{M}^{\text{In}}, \mathbf{M}^{\text{Out}} \in \mathbb{R}^{N\times N}$ denote weighted connections of outgoing and incoming edges in the RS-Graph, respectively. Considering a behavior sequence s=[(t_1, d_1,washing machine:start, laundry), (t_2, d_2, microwave:switch on, cooking), (t_3, d_1, washing machine:notification, laundry), (t_4, d_3, dryer:switch on, laundry), (t_5, d_2, microwave:notification, cooking), (t_6, d_4, dish washer: start, cooking)], the RS-Graph is shown in Figure 4(a) and the relevant incoming and outgoing matrices are shown in the Figure 4(b). Since several devices may appear in the behavior sequence repeatedly, we assign each edge a normalized weight calculated as the number of occurrences of the edge divided by the out-degree of the starting node of that edge.

4.2.2 Device Embedding Update on RS-Graph. To learn representations for devices, we first encode each device into a low-dimensional latent space. Let $e_d \in \mathbb{R}^L$ denote a L-dimensional embedding vector of device d. Let $\mathcal{N}_i^{(r)}$ denote the set of neighbor indices of node i under relation $r \in \mathscr{R}$. For nodes $j \in \mathcal{N}_i^{(r)}$, to evaluate how important node j's features are to node i, we compute the attention coefficient $E_{i,j,r}$ as:

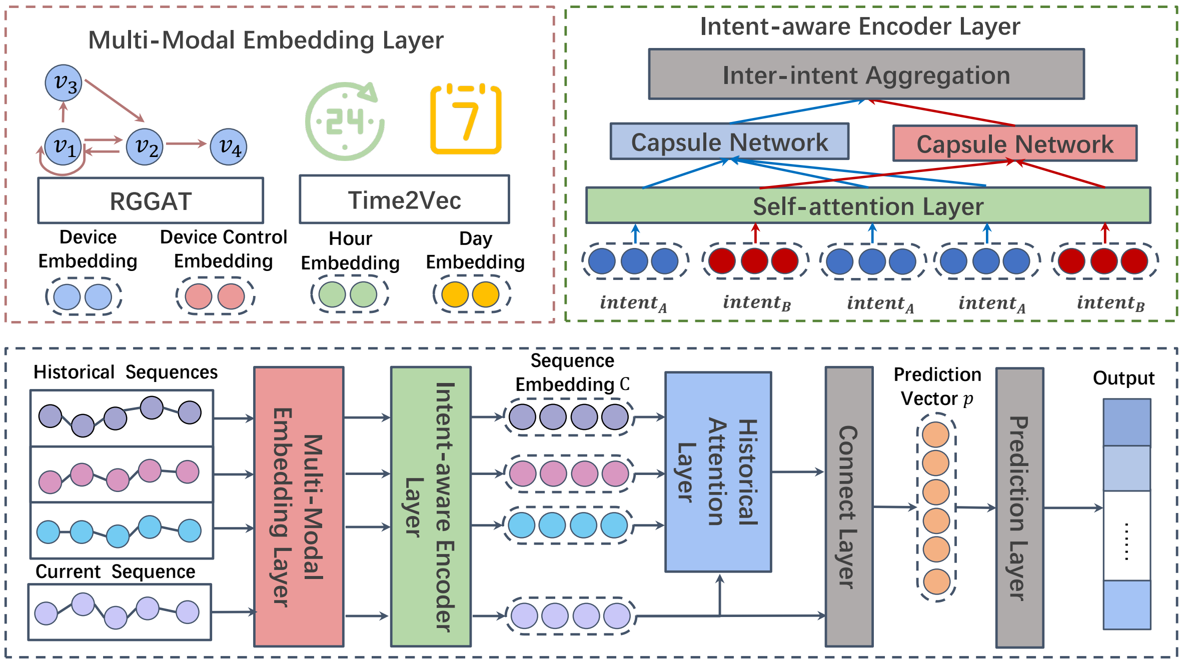
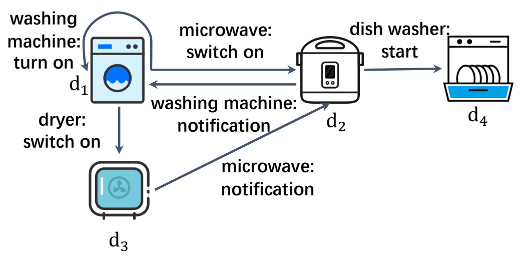
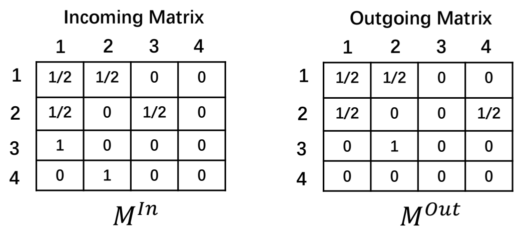


Figure 3: Overview of DeepUDI.



(a) An example of relational sequence graph.



Incoming Matrix

$$M^{In} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1/2 & 1/2 & 0 & 0 \\ 2 & 1/2 & 0 & 1/2 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \end{array}$$

Outgoing Matrix

$$M^{Out} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1/2 & 1/2 & 0 & 0 \\ 2 & 1/2 & 0 & 0 & 1/2 \\ 3 & 0 & 1 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{array}$$

(b) Incoming matrix $\mathbf{M}^{\text{In}}$ and the outgoing matrix $\mathbf{M}^{\text{Out}}$.

Figure 4: Construction of RSGraph.

$$E_{i,j,r} = \text{Attention}\left(\mathbf{W}\mathbf{e}_{d_i}, \mathbf{W}\mathbf{e}_{d_j}\right), \tag{2}$$

where $\mathbf{W}$ is the shared weight matrix, and Attention is the attention mechanism [24]. The normalized attention coefficients across all neighbors of node i under relation r are:

$$\alpha_{i,j,r} = \underset{j}{\text{softmax}}\left(E_{i,j,r}\right) = \frac{\exp\left(E_{i,j,r}\right)}{\sum_{k \in \mathcal{N}_i^{(r)}} \exp\left(E_{i,k,r}\right)}, \tag{3}$$

$$\forall i, r : \sum_{j \in \mathcal{N}_i^{(r)}} \alpha_{i,j,r} = 1. \tag{4}$$

Given the attention matrix $\mathbf{A}_r$ under relation r, where the value in row i and column j of $\mathbf{A}_r$ represents $\alpha_{i,j,r}$, and the connection matrices $\mathbf{M}^{\text{In}}$ and $\mathbf{M}^{\text{Out}}$, for the n-th device in the RS-Graph, the information propagation process can be formalized as:

$$\begin{aligned}\mathbf{a}_{n,r} = \text{Concat}\Big(&\mathbf{M}_n^{In} \odot \mathbf{A}_{r,n}\left[\mathbf{e}_{d_1}, \ldots, \mathbf{e}_{d_N}\right], \\ &\mathbf{M}_n^{Out} \odot \mathbf{A}_{r,n}\left[\mathbf{e}_{d_1}, \ldots, \mathbf{e}_{d_N}\right]\Big),\end{aligned} \tag{5}$$

where $\mathbf{M}_n^{In}, \mathbf{M}_n^{Out} \in \mathbb{R}^{1\times N}$ are n-th row of $\mathbf{M}^{In}$ and $\mathbf{M}^{Out}$ corresponding to node d_n, respectively, and $\mathbf{A}_{r,n} \in \mathbb{R}^{1\times N}$ is the n-th row of $\mathbf{A}_r$. $\odot$ denotes element-wise multiplication. $\mathbf{a}_{n,r}$ extracts the transition context information between different devices with different relations.

Then $\mathbf{a}_{n,r}$ is input to the Gate Recurrent Unit (GRU), which consists of an update gate $\mathbf{z}_n$ and a reset gate $\mathbf{r}_n$. The reset gate $\mathbf{z}_n$ determines how new input information is combined with previous memories

$$\mathbf{z}_{n,r} = \text{sigmoid}\left(\mathbf{W}_z \mathbf{a}_{n,r} + \mathbf{U}_z \mathbf{e}_{n-1}\right). \tag{6}$$

The update gate $\mathbf{r}_n$ determines what historical information to keep

$$\mathbf{r}_{n,r} = \text{sigmoid}\left(\mathbf{W}_r \mathbf{a}_{n,r} + \mathbf{U}_r \mathbf{e}_{n-1}\right). \tag{7}$$

Then, we constructs the candidate state $\tilde{\mathbf{e}}_{n,r}$ by the previous state e_{n-1}, the current state $\mathbf{a}_{n,r}$, and the reset gate $\mathbf{r}_{n,r}$ as

$$\tilde{\mathbf{e}}_{n,r} = \tanh\left(\mathbf{W}_h \mathbf{a}_{n,r} + \mathbf{U}_o\left(\mathbf{r}_{n,r} \odot \mathbf{e}_{n-1}\right)\right). \tag{8}$$

The final state is then the combination of the previous hidden state and the candidate state, under the control of the update gate. After updating all nodes in RS-Grpahs until convergence, we can obtain the device embedding $e_{n,r}$ under relation r as

$$\mathbf{e}_{n,r} = \left(1 - \mathbf{z}_{n,r}\right) \odot \mathbf{e}_{n-1} + \mathbf{z}_{n,r} \odot \tilde{\mathbf{e}}_{n,r}, \tag{9}$$

where $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_h \in \mathbb{R}^{L\times 2L}, \mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_o \in \mathbb{R}^{L\times L}$ are learnable parameters, $\odot$ represents element-wise multiplication. Adding the results of R relation outputs together can obtain the final n-th device embedding:

$$\mathbf{e}_d = \oplus_{r=1}^{R} \mathbf{e}_{n,r}, \tag{10}$$

where $\oplus$ represents element-wise addition.

The device control embedding $\mathbf{e}_c$ can be obtained similarly by building device control RS-Graphs, where the node represents device control and the incoming edge and the outgoing edge represent two relations, respectively.

4.3 Behavior Embedding

By concatenating the day of the week embedding, hour of the day embedding, device embedding and device control embedding, we can obtain the summarized representation $\tilde{\mathbf{h}}$ of each behavior:

$$\tilde{\mathbf{h}} = [\mathbf{e}_{dw}, \mathbf{e}_h, \mathbf{e}_d, \mathbf{e}_c], \tag{11}$$

To identify the position of the input variable, following [23], we add positional encoding PE to the behavior representation as follows:

$$\mathbf{h} = \tilde{\mathbf{h}} + PE, \tag{12}$$

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(pos/10000^{2i/d_{\tilde{\mathbf{h}}}}\right), \\ PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d_{\tilde{\mathbf{h}}}}\right), \end{aligned} \tag{13}$$

where i denotes the i-th dimension of the behavior embedding, pos denotes the position of the behavior in the behavior sequence and $d_{\tilde{\mathbf{h}}}$ is the dimension of $\tilde{\mathbf{h}}$.

5 INTENT-AWARE ENCODER LAYER

The purpose of the intent-aware encoder is to encode a sequence of behaviors while considering the relationship between different behaviors and the multiple intents of the user. To this end, we design a self-attention layer to mine the context of behaviors, and capsule networks to learn the multi-intent representations of users.

5.1 Self-attention Layer

We employ transformer encoder [23] for the self-attention layer since it can effectively mine global semantic information of behavior sequence context by learning query, key and value matrices of different variables. Given an input behavior representation $\mathbf{h}$, the query, key and value matrices can be calculated as following:

$$\mathrm{Q} = \mathbf{h}\mathrm{W}^Q,\ \mathrm{K} = \mathbf{h}\mathrm{W}^K,\ \mathrm{V} = \mathbf{h}\mathrm{W}^V, \tag{14}$$

where $\mathrm{W}^Q, \mathrm{W}^K, \mathrm{W}^V$ are the transformation matrices. The attention score is computed by:

$$\mathbf{A} = \mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{15}$$

where d_k is the dimension of K. To improve the stability of the learning process and achieve higher performance, we adopt multi-head attention in Q, K, and V. Then, the position-wise feed-forward network (FNN) and residual connections are adopted:

$$\overline{\mathbf{h}} = \mathrm{Trans}(\mathbf{h}) = \mathbf{h} + \mathbf{A}\mathbf{h} + \mathrm{FNN}(\mathbf{h} + \mathbf{A}\mathbf{h}), \tag{16}$$

where $\mathrm{Trans}(\cdot)$ is the transformer and $\mathrm{FNN}(\cdot)$ is a 2-layered position-wise feed-forward network [23].

5.2 Capsule Networks

We utilize Capsule Networks [19] (CapsNets) to extract multiple intents of the behavior sequence. A capsule is a set of neurons whose activity vectors represent the instantiated parameters of a specific type of entity, such as an object or object part, and the length of the instantiation vector represents the probability that a capsule's entity exists [19]. Take a two-layer capsule network as an example, there are two levels of capsules, i.e., low-level capsules from the first layer and high-level capsules from the second layer. The dynamic routing algorithm is used to compute the values of high-level capsules given the values of low-level capsules.

In DeepUDI, intent-specific behavior representations are treated as primary (low-level) capsules, while user multi-intent representations are treated as intent (high-level) capsules. We artificially classify the intents of behaviors, and then each behavior representation is only connected to the corresponding intent capsule. (As shown in Figure 3 (top right), the blue representations and the red representations belong to two different intents, and are connected to the corresponding intent capsules.) The representation $\overline{\mathbf{h}}_i$ of the i-th behavior denotes the i-th capsule of the primary layer. Based on the primary capsules, the j-th intent capsule of the next layer can be calculated as:

$$\hat{\mathbf{h}}_{j|i} = \mathbf{W}_{ij}\overline{\mathbf{h}}_i, \tag{17}$$

where $\mathbf{W}_{ij}$ denotes the learnable bilinear mapping matrix. The candidate vector $\mathbf{s}_j$ for intent capsule j is computed as the weighted sum of all primary capsules:

$$\mathbf{s}_j = \sum_i \mathbf{w}_{ij}\hat{\mathbf{h}}_{j|i}, \tag{18}$$

where w_{ij} denotes the weight for connecting low-level capsule i and high-level capsule j and is calculated by performing softmax on routing logits b_{ij} as:

$$\mathbf{w}_{ij} = \frac{\exp\left(b_{ij}\right)}{\sum_k \exp\left(b_{ik}\right)}, \tag{19}$$

where b_{ij} represents the log prior probability that capsule i should be coupled to capsule j. The values of b_{ij} are initialized to zeros, and updated by the routing process described in Algorithm 1. Then a non-linear "squashing" function [19] is adopted to squeeze the candidate vector s_j so that short vectors are compressed to almost zero length, while long vectors are compressed to a length slightly below 1. The vector of intent capsule j is calculated by:

$$\mathbf{c}_j = \text{squash}\left(\mathbf{s}_j\right) = \frac{\left\|\mathbf{s}_j\right\|^2}{1+\left\|\mathbf{s}_j\right\|^2}\frac{\mathbf{s}_j}{\left\|\mathbf{s}_j\right\|}. \tag{20}$$

The output intent capsules are formulated as $[\mathbf{c}_1, ..., \mathbf{c}_K] \in \mathbb{R}^{K \times len(\mathbf{c})}$ to represent multiple intents of the behavior sequences, the $len(\mathbf{c})$ denotes the dimension of intent capsule. Finally, the inter-intent aggregation mechanism is applied as follows:

$$\mathbf{C} = \mathbf{W}_C[\mathbf{c}_1, ..., \mathbf{c}_K], \tag{21}$$

where $\mathbf{W}_C \in \mathbb{R}^{1\times K}$ is the learnable parameter and denotes the weights of the intents.

Algorithm 1: DeepUDI Dynamic Routing.

```
Input: primary capsules $\overline{\mathbf{h}}_i$, iteration times $T$, number of intent
    capsules $K$
Output: intent capules $\{\mathbf{c}_j, j = 1, \dots, K\}$
1: for each primary capsule $i$ and corresponding intent capsule $j$:
   initialize $b_{ij} = 0$
2: for $iter = 1, ...T$ do
3:     for each primary capsule $i$: $\mathbf{w}_i$ = softmax $(\mathbf{b}_i)$.
4:     for each intent capsule $j$: $\mathbf{s}_j = \sum_i \mathbf{w}_{ij}\mathbf{W}_{ij}\mathbf{h}_i$.
5:     for each intent capsule $j$: $\mathbf{c}_j$ = squash $(\mathbf{s}_j)$.
6:     for each primary capsule $i$ and intent capsule $j$:
       $b_{ij} = b_{ij} + \mathbf{c}_j^\top \mathbf{W}_{ij}\mathbf{h}_i$.
7: end for
8: return $\{\mathbf{c}_j, j = 1, \dots, K\}$
```

6 HISTORICAL ATTENTION LAYER

[20] shows the multi-level nature of human behavior periodicity. However, a static representation of user behavior sequences fails to capture the periodicity and evolution of user behavior. In DeepUDI, we propose a historical attention layer to enable the model to mine periodic features from the up-to-date historical behavior sequences.

The historical attention layer uses the representation $\mathbf{C}_t$ of the current behavior sequence as a query vector to calculate the attention score between it and the historical behavior sequences. The historical attention mechanism is formulated as follows:

$$\alpha_i = \frac{\exp\left(\beta_i\right)}{\sum_{j=1}^{t-1}\exp\left(\beta_j\right)}, \quad \beta_i = \tanh\left(\mathbf{C}_t\mathbf{W}_H\mathbf{C}_i\right), \tag{22}$$

where $\alpha_i, \beta_i \in \mathbb{R}$ are normalized and unnormalized scores for the i-th history behavior sequence, respectively, $\mathbf{C}_i$ is the i-th history behavior sequence's representation and W_H is the learnable parameter. Upon obtaining the attention weights, the prediction vector p can be obtained by concating $\mathbf{C}_t$ and the weighted sum of historical behavior sequence representations:

$$p = \text{Concat}\left(\mathbf{C}_t, \sum_{i=1}^{t-1}\alpha_i\mathbf{C}_i\right), \tag{23}$$

Table 1: Datasets Description

Name	Time period (Y-M-D)	Sizes	# Devices	# Device controls
US	2022-02-22~2022-03-21	67,882	40	268
SP	2022-02-28~2022-03-30	15,665	34	234
FR	2022-02-27~2022-03-25	4,423	33	222
AN	2022-07-31~2022-08-31	1,765	36	141

Finally, we feed p into the prediction layer and compute the probabilities of device controls as follows:

$$\hat{\mathbf{y}} = \text{softmax}\left(\mathbf{W}_p p\right), \tag{24}$$

where $\hat{\mathbf{y}}$ is the predicted probabilities of the next device control and $\mathbf{W}_p \in \mathbb{R}^{|C|\times len(p)}$ is the learnable transformation matrix, $|C|$ is the number of device controls, $len(p)$ is the length of p.

7 EXPERIMENTS

In this section, we conduct comprehensive experiments on four real-world datasets to answer the following research questions:

- RQ1. **Performance.** Compared with other methods, does DeepUDI have higher prediction performance of user device interaction?
- RQ2. **Ablation study.** How do main components of DeepUDI affect the performance of UDI prediction?
- RQ3. **Parameter study.** How key parameters affect the performance of DeepUDI?
- RQ4. **Interpretability study.** Can DeepUDI give a reasonable explanation for the prediction results?

7.1 Experimental Setup

7.1.1 Datasets. We evaluate model performance using four real-world smart home datasets, three (FR/SP/US) from public datasets[1] and one anonymous dataset (AN) collected by ourselves. The datasets description is shown in Table 1. All datasets are split into training, validation and testing sets with a ratio of 7:1:2. We label user behaviors with intents based on device attributes. For example, oven belongs to cooking intent, and TV belongs to entertainment intent.

7.1.2 Baselines. We compare DeepUDI with existing user device interaction prediction schemes in smart home and sequential user behavior prediction schemes.

- **HMM** [10] regards all the device controls as states and builds a transition matrix to capture the first order transition probabilities between them.
- **FPMC** [18] combines markov chain with matrix factorization to capture both sequential patterns and user preferences for UDI prediction.
- **LSTM** [22] captures the long-term sequential influence, and it can be applied to UDI prediction.
- **CA-RNN** [12] uses context-specific transition matrix in RNN cell to consider context-dependent features in a sequential recommendation.

[1] https://github.com/snudatalab/SmartSense

- **DeepMove** [4] can be regarded as an enhanced version of RNN with history attention mechanism. It captures both user long and short-term mobility patterns.
- **SR-GNN** [25] applies gated graph neural network to generate latent vectors of items and then represents each session through attention network for user behavior prediction.
- **SmartSense** [6] applies query transformer and commonsense knowledge for smart home action recommendation.

7.1.3 Evaluation metrics. As UDI prediction is a multi-classification problem, we use two metrics, top-k accuracy (Acc@K) and Macro-F1, to evaluate the performance.

7.1.4 Implementation. All models (including baselines and DeepUDI) are implemented by PyTorch [16] and run on a graphic card of GeForce RTX 3090 Ti. All models are trained with Adam optimizer [9] with learning rate 0.001 and l_2 regularization coefficient 0.00001. We train DeepUDI to minimize the cross-entropy loss. During training, we monitor Acc@1 and stop training if there is no performance improvement in 10 steps. To ensure the efficiency of training, we consider a fixed number of the up-to-date historical sequences rather than all.

7.2 Performance Comparison (RQ1)

We use grid search to adjust the parameters (§7.4) of DeepUDI and select the optimal results. The results are shown in Table 2. Bold values indicate the optimal performance, and underlined values indicate the second best performance. As can be observed, the proposed DeepUDI scheme outperforms all competitors in most cases. This is because our model simultaneously considers the complex transitions, users' multi-intent, and multi-level periodicity. The traditional models HMM and FPMC show the worst performance. The RNN-based models outperform the traditional models because of stronger sequence modeling capability. SR-GNN outperforms RNN-based models, because it can model complex transitions in behavior sequences. By exploiting transformer to mine contextual information, SmartSense achieves better performance than all other baselines, but is still inferior to our proposed scheme.

7.3 Ablation Study (RQ2)

DeepUDI has three main components, Multi-Modal Embedding Layer (MME), Intent-aware Encoder (IAE) and Historical Attention Mechanism (HAM). To verify the contribution of each component to the final prediction results, we conduct ablation experiments while keeping the optimal parameters unchanged. DeepUDI-MME denotes DeepUDI without MME layer, which, instead of RGGAT and Time2Vec, uses a simple embedding layer (i.e., a fully connected layer). DeepUDI-IAE denotes DeepUDI without IAE layer. DeepUDI-HAM denotes DeepUDI without HAM layer. DeepUDI-ALL denotes a DeepUDI with none of the three components. DeepUDI is the scheme with all the three components. Table 3 shows that DeepUDI outperforms all others on both AN and FR datasets, while DeepUDI-ALL shows the worst Acc@1 and Macro-F1. In summary, each of the three components of DeepUDI is helpful for UDI prediction.

7.4 Influence of Hyper-parameters (RQ3)

7.4.1 Number of layers in RGGAT. Figure 5(a) shows the performance of DeepUDI with different numbers of RGGAT layers. We find that when the number of RGGAT layers increases, Acc@1 will first increase and then decrease, reaching the optimal value at 2 layers. This is because fewer layers can lead to under-fitting, and too many layers can lead to over-smoothing [3].

7.4.2 Embedding dimension. As shown in Figure 5(b), when the embedding dimension is too small, it cannot encode enough information, resulting in poor performance. As the embedding dimension becomes larger, the performance will gradually increase. A larger embedding size does not necessarily lead to better performance because of over-fitting issue. Therefore, we choose the embedding size to be 50 to achieve the best performance.

7.4.3 Number of history behavior sequences. A larger number of historical sequences allows the model to consider more historical information. However, too much historical information may introduce more uncertainty and not contribute to performance improvement. Figure 5(c) shows that 15 historical sequences enables DeepUDI to achieve the optimal performance.

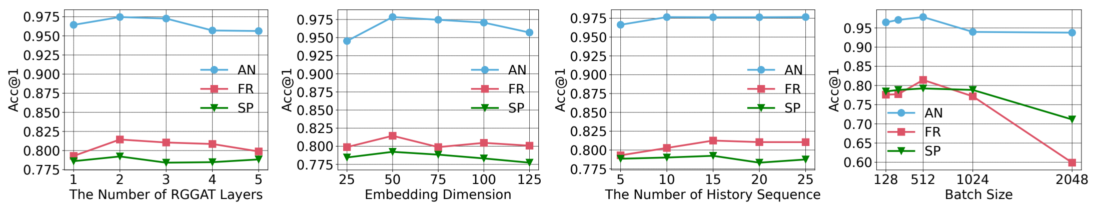
7.4.4 Batch Size. Figure 5(d) shows the influences of batch size. As the batch size increases, Acc@1 increases, and when the batch size exceeds 512, the increase in batch size leads to a decrease in performance due to larger batch size lower the generalization ability of the model.

7.5 Case Study (RQ4)

To verify the interpretability of DeepUDI, we choose a behavior sequence from the test set of the AN dataset and visualize the attention weight of the RGGAT and the multi-intent weights in the inter-intent aggregation layer. From the results shown in 6, two observations can be made. First, RGGAT can dig out potential device correlations from complex and heterogeneous device transitions, which reflect the user's behavior habits. The high correlation between sweeper, window cleaner and TV shows that the user tends to do some entertainment activities while doing housework activities. The correlation between curtains and bedlight is high because both are operated right before bedtime and after getting up. Second, DeepUDI can well explain the prediction results based on the intent weight in the inter-intent aggregation layer. From the intent weight heatmap, we can find that the four intents with the highest weights in the capsule network are "sleep/getup", "shower", "leave/return", and "entertainment", which reasonably explains why "bedlight:off" (sleep/getup) , "shower:on" (shower), "air conditioner:change" (others) "TV:off" (entertainment) are predicted by the model to be the next action with the highest probabilities. In particular, the next real action of the sequence is "bedlight:off", which is also predicted with the highest probability by the model. The reason why the predicted probabilities of "shower:on" and "air conditioner:change" are also relatively high is that the model has learned that the user often takes a shower and adjusts the temperature of the air conditioner before going to bed. The high probability of "TV:off" in the prediction result is because the user had already turned on the TV and needs to turn it off some time before bed.

Table 2: Performance comparison on four real world datasets.

Dataset	Metric	HMM	FPMC	LSTM	CA-RNN	DeepMove	SRGNN	SmartSense	DeepUDI(Ours)
AN	Acc@1	0.6099	0.6557	0.7062	0.7026	0.7116	0.9245	<u>0.9407</u>	**0.9784**
	Acc@3	0.7501	0.7959	0.7843	0.8302	0.9272	<u>0.9864</u>	0.9731	**0.9865**
	Acc@5	0.7714	0.7902	0.8328	0.9003	0.9542	<u>0.9872</u>	0.9838	**0.9892**
	Macro-F1	0.2439	0.2845	0.3759	0.4159	0.5027	0.7368	<u>0.7519</u>	**0.7997**
FR	Acc@1	0.6536	0.6814	0.6962	0.7893	0.7762	0.7819	<u>0.7923</u>	**0.8144**
	Acc@3	0.7813	0.8271	0.8011	0.9148	0.9221	0.9197	**0.9371**	<u>0.9238</u>
	Acc@5	0.8242	0.8508	0.8565	0.9425	0.9446	0.9435	**0.9628**	<u>0.9512</u>
	Macro-F1	0.1127	0.1279	0.1302	0.2102	0.2288	0.2482	<u>0.2603</u>	**0.3425**
SP	Acc@1	0.6315	0.6964	0.7517	0.7853	0.7756	0.7815	<u>0.7921</u>	**0.7923**
	Acc@3	0.7863	0.7916	0.8864	0.8915	0.9125	0.9303	<u>0.9342</u>	**0.9375**
	Acc@5	0.8361	0.8605	0.9346	0.9117	0.9521	<u>0.9603</u>	0.9511	**0.9642**
	Macro-F1	0.1382	0.1586	0.1756	0.1745	0.2159	0.2239	<u>0.2244</u>	**0.3112**
US	Acc@1	0.3327	0.3543	0.4286	0.5212	0.5527	0.5784	<u>0.5935</u>	**0.6056**
	Acc@3	0.6881	0.6992	0.8209	0.8577	0.8844	0.8955	<u>0.9056</u>	**0.9123**
	Acc@5	0.7258	0.7712	0.8929	0.9135	0.9418	0.9463	<u>0.9489</u>	**0.9521**
	Macro-F1	0.1069	0.1123	0.1265	0.1396	0.2388	0.2431	<u>0.2451</u>	**0.3538**



(a) Number of layers of RGGAT.　(b) Embedding dimension.　(c) Number of history behavior sequences.　(d) Batch size.

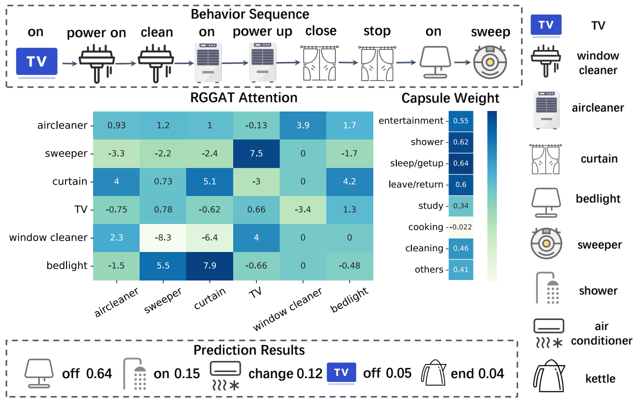
Figure 5: The influence of hyper-parameters on Acc@1.

Table 3: The Acc@1 and Macro-F1 results on AN and FR.

Model	AN		FR	
	Acc@1	Macro-F1	Acc@1	Macro-F1
DeepUDI-MME	0.9487	0.7364	0.7851	0.2805
DeepUDI-IAE	0.9595	0.7585	0.7853	0.3103
DeepUDI-HAM	0.9676	0.7812	0.7988	0.3234
DeepUDI-ALL	0.9137	0.6934	0.7578	0.2511
DeepUDI	**0.9784**	**0.7997**	**0.8144**	**0.3425**

8 CONCLUSION

In this paper, we propose DeepUDI, a novel user device interaction prediction framework, which achieves accurate UDI prediction by addressing three challenges: 1) complex heterogeneous transitions; 2) multiple intents of users; and 3) multi-level periodicity of user behaviors. Specifically we model user's behavior sequences as RS-Graphs, and RGGAT is proposed to mine the complex heterogeneous transitions in behavior sequences. We propose an Intent-aware Encoder (IAE), which consists of capsule networks and inter-intent aggregation mechanism to learn user multi-intent representations. To learn the multiple periodicity of user behaviors, we propose a Historical Attention Mechanism (HAM) to learn periodic behavior representations from historical behavior sequences. Comprehensive experiments on four real-world datasets demonstrate the effectiveness and interpretability of DeepUDI.



	aircleaner	sweeper	curtain	TV	window cleaner	bedlight
aircleaner	0.93	1.2	1	-0.13	3.9	1.7
sweeper	-3.3	-2.2	-2.4	7.5	0	-1.7
curtain	4	0.73	5.1	-3	0	4.2
TV	-0.75	0.78	-0.62	0.66	-3.4	1.3
window cleaner	2.3	-8.3	-6.4	4	0	0
bedlight	-1.5	5.5	7.9	-0.66	0	-0.48

Figure 6: Attention score of RGGAT, capsule weight and top 5 prediction results of the example.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive feedback and comments. This work is supported by the National Key Research and Development Program of China under grant No. 2020YFB1804704, the National Natural Science Foundation of China under grant No. 61972189, the Major Key Project of PCL under grant No. PCL2021A03-1, Shenzhen Science and Technology Innovation Commission: Research Center for Computer Network (Shenzhen) Ministry of Education, and the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959989.

REFERENCES

[1] Khaled A. Alaghbari, Mohamad Hanif Md. Saad, Aini Hussain, and Muhammad Raisul Alam. 2022. Activities Recognition, Anomaly Detection and Next Activity Prediction Based on Neural Networks in Smart Homes. *IEEE Access* 10 (2022), 28219–28232. https://doi.org/10.1109/ACCESS.2022.3157726

[2] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. 2019. Relational graph attention networks. *arXiv preprint arXiv:1904.05811* (2019).

[3] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3438–3445.

[4] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proceedings of the 27th International Conference on World Wide Web (WWW)*. ACM, Lyon, 1459–1468.

[5] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. 2012. Next place prediction using mobility markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*. ACM, 1–6.

[6] Hyunsik Jeon, Jongjin Kim, Hoyoung Yoon, Jaeri Lee, and U Kang. 2022. Accurate Action Recommendation for Smart Home via Two-Level Encoders and Commonsense Knowledge. In *Proceedings of the 31th ACM International Conference on Information & Knowledge Management (CIKM)*. ACM, Atlanta, Georgia, USA, 1–10.

[7] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE 18th International Conference on Data Mining (ICDM)*. IEEE, Singapore, 197–206.

[8] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. 2019. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321* (2019).

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Miao Lin, Vincent W. Zheng, and Shili Xiang. 2018. Sequential context modeling for smart devices by Collaborative Hidden Markov Model. In *4th IEEE World Forum on Internet of Things, WF-IoT 2018, Singapore, February 5-8, 2018*. IEEE, 771–777. https://doi.org/10.1109/WF-IoT.2018.8355155

[11] Liwei Liu, Wei Chen, Lu Liu, Kangkang Zhang, Jun Wei, and Yan Yang. 2021. TAGen: Generating Trigger-Action Rules for Smart Homes by Mining Event Traces. In *Service-Oriented Computing - 19th International Conference, ICSOC 2021, Virtual Event, November 22-25, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 13121)*, Hakim Hacid, Odej Kao, Massimo Mecella, Naouel Moha, and Hye-young Paik (Eds.). Springer, 652–662. https://doi.org/10.1007/978-3-030-91431-8_41

[12] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, Barcelona, Spain, 1053–1058.

[13] Knud Lasse Lueth. 2018. State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating. https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[14] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. *Advances in Neural Information Processing Systems (NIPS)* 20 (2007).

[15] Mustafa A. Mustafa, Alexandros Konios, and Matias Garcia-Constantino. 2021. IoT-Based Activities of Daily Living for Abnormal Behavior Detection: Privacy Issues and Potential Countermeasures. *IEEE Internet Things Mag.* 4, 3 (2021), 90–95. https://doi.org/10.1109/IOTM.0001.2000169

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NIPS)* 32 (2019).

[17] Lakshmanan Rakkappan and Vaibhav Rajan. 2019. Context-aware sequential recommendations withstacked recurrent neural networks. In *Proceedings of the 28th International Conference on World Wide Web (WWW)*. ACM, San Francisco, 3172–3178.

[18] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*. ACM, Raleigh, NC, USA, 811–820.

[19] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. *Advances in Neural Information Processing Systems (NIPS)* 30 (2017).

[20] Vijay Srinivasan, Christian Koehler, and Hongxia Jin. 2018. RuleSelector: Selecting conditional action rules from user behavior patterns. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)* 2, 1 (2018), 1–34.

[21] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, Los Angeles, California, USA, 565–573.

[22] Niek Tax. 2018. Human Activity Prediction in Smart Home Environments with LSTM Neural Networks. In *14th International Conference on Intelligent Environments, IE 2018, Roma, Italy, June 25-28, 2018*. IEEE, 40–47. https://doi.org/10.1109/IE.2018.00014

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems (NIPS)* 30 (2017).

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *Proceedings of the 6th International Conference on Learning Representations (ICLR)* (2018).

[25] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on Artificial Intelligence*, Vol. 33. AAAI, Hilton Hawaiian Village, Honolulu, Hawaii, USA, 346–353.

[26] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation.. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, Vol. 19. Morgan Kaufmann, Macao, China, 3940–3946.

[27] Masaaki Yamauchi, Masahiro Tanaka, Yuichi Ohsita, Masayuki Murata, Kensuke Ueda, and Yoshiaki Kato. 2021. Smart-home anomaly detection using combination of in-home situation and user behavior. *CoRR* abs/2109.14348 (2021). https://arxiv.org/abs/2109.14348