I Know Your Intent: Graph-enhanced Intent-aware User Device Interaction Prediction via Contrastive Learning

JINGYU XIAO* and QINGSONG ZOU*, Tsinghua Shenzhen International Graduate School, China and Peng Cheng Laboratory, China
QING LI† and DAN ZHAO, Peng Cheng Laboratory, China
KANG LI, Jilin University, China
ZIXUAN WENG, Beijing Jiaotong University, China
RUOYU LI and YONG JIANG, Tsinghua Shenzhen International Graduate School, China and Peng Cheng Laboratory, China

With the booming of smart home market, intelligent Internet of Things (IoT) devices have been increasingly involved in home life. To improve the user experience of smart homes, some prior works have explored how to use machine learning for predicting interactions between users and devices. However, the existing solutions have inferior User Device Interaction (UDI) prediction accuracy, as they ignore three key factors: routine, intent and multi-level periodicity of human behaviors. In this paper, we present SmartUDI, a novel accurate UDI prediction approach for smart homes. First, we propose a Message-Passing-based Routine Extraction (MPRE) algorithm to mine routine behaviors, then the contrastive loss is applied to narrow representations among behaviors from the same routines and alienate representations among behaviors from different routines. Second, we propose an Intent-aware Capsule Graph Attention Network (ICGAT) to encode multiple intents of users while considering complex transitions between different behaviors. Third, we design a Cluster-based Historical Attention Mechanism (CHAM) to capture the multi-level periodicity by aggregating the current sequence and the semantically nearest historical sequence representations through the attention mechanism. SmartUDI can be seamlessly deployed on cloud infrastructures of IoT device vendors and edge nodes, enabling the delivery of personalized device service recommendations to users. Comprehensive experiments on four real-world datasets show that SmartUDI consistently outperforms the state-of-the-art baselines with more accurate and highly interpretable results.

CCS Concepts: • **Human-centered computing → Human computer interaction (HCI)**.

Additional Key Words and Phrases: Smart Home;Human Device Interaction; Graph Neural Networks; Contrastive Learning

ACM Reference Format:
Jingyu Xiao, Qingsong Zou, Qing Li, Dan Zhao, Kang Li, Zixuan Weng, Ruoyu Li, and Yong Jiang. 2023. I Know Your Intent: Graph-enhanced Intent-aware User Device Interaction Prediction via Contrastive Learning. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 3, Article 136 (September 2023), 28 pages. https://doi.org/10.1145/3610906

136

*The first two authors have equal contribution.
†Corresponding Author: Qing Li (liq@pcl.ac.cn).

Authors' addresses: Jingyu Xiao, jy-xiao21@mails.tsinghua.edu.cn; Qingsong Zou, zouqs21@mails.tsinghua.edu.cn, Tsinghua Shenzhen International Graduate School, Shenzhen, China and Peng Cheng Laboratory, Shenzhen, China; Qing Li, liq@pcl.ac.cn; Dan Zhao, zhaod01@pcl.ac.cn, Peng Cheng Laboratory, Shenzhen, China; Kang Li, Jilin University, Changchun, China, likang9920@mails.jlu.edu.cn; Zixuan Weng, Beijing Jiaotong University, Beijing, China, 20722027@bjtu.edu.cn; Ruoyu Li, liry19@mails.tsinghua.edu.cn; Yong Jiang, jiangy@sz.tsinghua.edu.cn, Tsinghua Shenzhen International Graduate School, Shenzhen, China and Peng Cheng Laboratory, Shenzhen, China.

2474-9567/2023/9-ART136 $15.00
https://doi.org/10.1145/3610906

1 INTRODUCTION

With fast-evolving IoT solutions, the number of smart devices in homes has soared, expected to reach 5 billion by 2025 [24]. The emergence of cloud platforms also allows IoT sensors and actuators to better assist users in various home living activities [19, 20]. User Device Interaction (UDI), i.e., constant device controls, can reflect users' behavioral habits and intents. UDI prediction for smart homes brings about opportunities from multiple perspectives. For service providers, such as vendors, predicting users' living behaviors through their device usage histories can offer insights for improving user experience. From the perspective of device intelligence, prediction of users' behaviors can help intelligent platforms recommend actions that users may like to perform, such as "*turn off the bed light*", as shown in Fig 1. From the perspective of user behavior analysis, accurate user behavior prediction can be used for abnormal user behavior identification, elderly/disabled care, or further user behavior analysis.

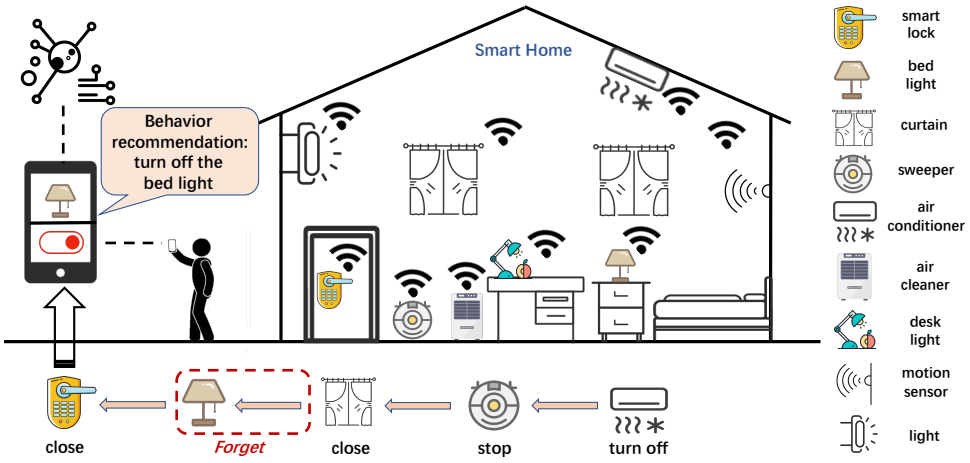


Fig. 1. The user forgets to turn off the bed light after leaving, by analyzing the behavior sequence, the user behavior prediction model recommends the action "*turn off the bed light*".

Motivated by these, a growing body of research endeavors to propose algorithms that harness the interaction data between smart home users and devices to empower users with intelligent services that go beyond mere device interactions. These services encompass tailored recommendations for device behavior, ensuring a seamless and personalized experience within the smart home ecosystem. Some prior studies have analyzed user behaviors in smart home and adopted user behavior prediction technologies for different purposes. [22, 32] infer the users' behavioral intents from their sequential operations on the device, and generate alternative automation rules and subsequent behavior recommendations. [1, 25, 41] 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. [13] applies deep learning models to recommend actions for users to control their devices at smart home.

However, user behavior is dynamic and complex [39], making User Device Interaction (UDI) prediction more challenging. There are three key factors in the prediction of UDI that have not been properly considered in smart home scenarios.

First, a **routine** contains people's behavior habits. Mining the correlations of behaviors in routines is beneficial for UDI prediction. However, the existence of noise behaviors between the routine behaviors causes the model to learn false correlations between noisy behaviors and routine behaviors which co-occur in the same sequence.

For example, as shown in Fig. 2, a user's routine is to cook while doing laundry. Meanwhile, the user also has random behaviors in the sequence, such as "*unlock the smart lock*" at time t_3, and "*open the network audio*" at time t_{10}. The correlation between laundry and cooking behaviors should be relatively high while "*unlock the smart lock*" and "*open the network audio*" should be less correlated with former behaviors, however, since all the above behaviors appear in the same sequence, it is difficult for the model to distinguish the strength of the correlation between different behaviors. The challenge of learning user routines is to identify the routines and learn the correct correlation of behaviors from the same and different routines.

Second, an **intent** inherently determines the user behaviors. Relying purely on a user's sequential behavior without considering intents may lead to wrong predictions. On the one hand, there are often multiple intents in a UDI sequence. For example, a user may cook while doing laundry because of the long wash cycle of the washing machine, as shown in Fig. 2. As such, there are two *intents*, laundry and cooking, in the behavior sequence. Without considering a user's multiple intents, the next behavior after observing t_1 to t_5 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 user's next behavior should be "*turn off washing machine*" as the washing machine cycle has finished. On the other hand, user intents are dynamic. There are complex transitions between different intents, which leads to complex transitions between devices user accessed. 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 Fig. 2, the user interacts with the water valve at time t_1 and t_7. All devices between time t_1 and t_7, rather than just the devices immediately next to t_1 and t_7, have transitions with the water valve. Meanwhile, 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). The challenge of learning user intents lies in reasonably mining the intents behind user behaviors and properly modeling complex heterogeneous transitions between behaviors.

Third, **multi-level periodicity** in user behaviors 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, can fluctuate within a week. As shown in Fig. 3, the user may work overtime and go to bed later than usual once every weekend, which shows week-level periodicity. As a mountaineering enthusiast, the user goes climbing at the end of each month and goes to bed early at night due to fatigue, which shows month-level periodicity. In this example, the multi-level periodicity in the user's behaviors, i.e., leaving work on time daily, working overtime every week, and going climbing every month, causes fluctuations in sleeping time. The challenge of modeling multi-level periodicity lies in accurately and efficiently capturing the correlation between the current behavior sequence and historical behavior sequences.

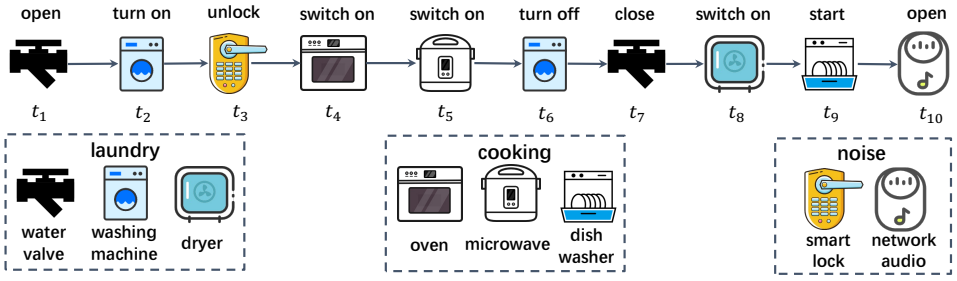


Fig. 2. An example of user sequential behavior with two intents: laundry (watervalve/washing machine/dryer) and cooking (oven/microwave/dish washer). Noise behaviors include unlock the smart lock and open the network audio.

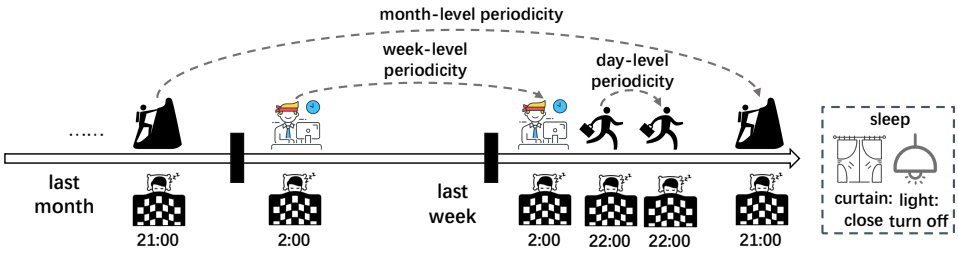


Fig. 3. The multi-level periodicity in the user behaviors.

Although some existing models such as CNN [33], RNN [27], GNN [38] and Transformer [13] models can model temporal information and mine contextual information well, they cannot be directly applied to smart home scenarios due to the following reasons. First, these models treat the entire sequence indiscriminately, therefore learns wrong correlations between behaviors due to lack of identification of **routines**. Second, these models do not consider **intents** of the user, thus lacking of capturing the complex heterogeneous transitions between user behaviors. Third, these models only consider the current sequence. So they naturally cannot mine the correlation between the current sequence and the historical sequences, which prevents them from modeling **multi-level periodicity**.

To address the above challenges, in this paper, we propose SmartUDI, a novel approach for accurate UDI prediction. The design of SmartUDI mainly includes the following three key designs. First, to make the model learn the correct behavioral correlation in routines, SmartUDI designs a **M**essage-**P**assing-based **R**outine **E**xtraction (**MPRE**) to extract routine behavior from behavior sequences with noise behaviors, then contrastive learning is applied to minimize the difference between behaviors within the same routine and maximize the difference between the behaviors derived from different routines. Second, we propose an **I**ntent-aware **C**apsule **G**raph **A**ttention Ne**T**work (**ICGAT**) to encode multiple intents of users while considering complex heterogeneous transitions. Specifically, ICGAT mines transitions among devices by relational gated graph attention network and views behaviors of different intents as different primary capsules, and learns multi-intent representations of users through the capsule networks [29], then an inter-intent aggregation mechanism is applied to learn weights of different intents for aggregating representations. Third, we design a **C**luster-based **H**istorical **A**ttention **M**echanism (**CHAM**) to capture the multi-level periodicity by aggregating the current sequence and the historical sequence representations through the attention mechanism. To take advantage of effective historical sequences and improve training efficiency, we cluster the behavior sequences and leverage semantically nearest historical sequences for aggregation. Our main contributions and novelties are summarized as follows:

- We propose SmartUDI, an innovative framework designed to enhance the accuracy of User Device Interaction (UDI) prediction by considering crucial factors such as user routines, intents, and multi-level periodicity. SmartUDI can be seamlessly deployed on cloud infrastructures of IoT device vendors and edge nodes, allowing for comprehensive analysis of user behavior data and extraction of valuable behavior patterns. By leveraging this intelligent framework, SmartUDI enables the delivery of personalized device service recommendations to users, thereby augmenting the overall intelligence and efficiency of the home Internet of Things ecosystem.
- We propose **MPRE** to mine routine behaviors from behavior sequence with noisy and apply contrastive learning to correct correlations between behaviors. Specially, we are the *first* to formalize the routine classification problem as a community detection problem and design a *novel* message-passing-based routine classification algorithm to solve it.

- We propose **ICGAT**, an architecture based on graph attention networks, which converts user behavior sequences into relational sequence graphs to learn complex heterogeneous transitions among devices and view behaviors as primary capsules to learn multi-intent representations of users by dynamic routing. We *innovatively* propose to construct RS-Graph from behavior sequence and use the relational gated graph attention network to learn the heterogeneous transitions between behaviors on RS-Graph. Besides, we explicitly label the intents of the behavior and use a dynamic routing algorithm to realize the intent learning. *All of these have not been explored in prior works.*
- We propose **CHAM** to capture multi-level periodicity of user behaviors by applying attention mechanism between current and the semantically nearest historical user behavior sequences. Particularly, we propose a *novel* measure of behavior sequence similarity which jointly considers the semantic and temporal features of behavior sequences. Moreover, we are the *first* to apply clustering to improve historical attention mechanism, CHAM can filter out noisy behavior sequences and improve computational efficiency by performing attention on the most similar historical sequences.
- We conduct comprehensive experiments on four real-world datasets including three open-sourced datasets and one newly collected datasets from our testbed. The results show that SmartUDI consistently outperforms state-of-the-art baselines and shows better interpretability.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 describes the problem definition. Section 4 introduces the overview of SmartUDI architecture. Section 5, 6, 7 respectively introduce the important components Message-Passing-based Routine Extraction, Intent-aware Capsule Attention Network and Cluster-based History Attention Mechanism of SmartUDI. Section 8 describes the multi-task training method. Section 9 shows experimental results. Section 11 concludes the whole paper.

2 RELATED WORK

2.1 User Behavior Prediction in Smart Homes

Research on user behavior prediction in smart homes mainly focuses on learning-based methods, namely traditional machine learning methods and deep learning methods. (1) **Traditional methods.** Hidden Markov Model (HMM) [2, 31] is applied to extract the user's behavior pattern from the user's interaction behavior sequence with the devices, and detect the user's abnormal behaviors. However, when facing behavioral sequences with multiple intents and periodic sequential patterns, HMM will fails due to that the independence assumption [7] of HMM makes it unable to consider context information. (2) **Deep learning methods.** Some works try to apply deep learning models to achieve better performance. [1, 6, 8, 34, 42] exploit Long Short-Term Memory Networks (LSTM) to predict user behavior in smart home scenarios. However, LSTM can only model long-term sequential influence [10], but misses out the complex heterogeneous transitions and periodic sequential patterns caused by users' routines, intents and multi-level periodicity. SmartSense [13] adopts knowledge transfer to exploit user intent and employs a two-stage encoder to mine contextual information. However, it has a few drawbacks. First, it does not classify the routines before inputting the sequence into the encoder. Second, the complex heterogeneous transitions are not modeled. Third, it does not take into account multi-level periodicity due to lack of consideration of historical sequences. DeepUDI [39] employs the attention mechanism to consider the periodicity of human behavior and intent-aware encoder to consider intents. However, one the one hand, it does not consider routines, on the other hand, it leads to suboptimal performance and long prediction time since it ambiguously considers all history sequences. In conclusion, none of the above works can consider routines, intents and multi-level periodicity at the same time in UDI sequences, which leads to inferior UDI prediction accuracy.

2.2 Sequential User Behavior Prediction

Sequential user behavior prediction methods can be categoried as follows. (1) **Traditional methods.** Factorizing Personalized Markov Chains (FPMC) [28] factorizes the users-locations matrix to generate user general preferences to complete next location prediction. However, the location independence assumption prevents FPMC from capturing the complicated sequential information [28]. (2) **CNN-based methods.** Caser [33] employs CNN in both time-axis and feature-axis to capture temporal dynamics in sequential recommendation. However, due to the limited receptive field of CNN, it cannot fully model long-term contextual information [17].(3) **RNN-based methods.** CARNN [23] and SIAR [27] incorporate contextual information into the RNN for sequential recommendation. However, CARNN and SIAR can not model multi-level periodicity because they only mine contextual information in a single sequence [23, 27]. Although DeepMove [9] uses attention-based RNN to model the correlation between the current sequence and historical sequences, it leads to suboptimal performance and long prediction time since it ambiguously considers all history sequences. (4) **GNN-based methods.** SRGNN [38, 40] applies gated GNN to capture complex transition patterns among nodes for session-based recommendation. However, SRGNN ignores the heterogeneity of the transition patterns caused by users' intents [38]. (5) **Transformer-based methods.** SASRec [14] utilizes unidirectional transformers to capture sequential patterns in sequences while considering the importance of correlations between behaviors. However, SASRec only captures the sequential patterns in single sequences [35], which prevents it from considering multi-level periodicity.

3 PROBLEM FORMULATION

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 $b = (t, d, dc, i)$, is a 4-tuple consisting of time t, device $d \in \mathcal{D}$, device control $dc \in C$, and intent $i \in \mathcal{I}$. For example, 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.*

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.

4 SMARTUDI OVERVIEW

To consider user routines, intents and multi-level periodicity, we propose SmartUDI, depicted in Fig. 4. SmartUDI mainly consists of a Message-Passing-based Routine Extraction algorithm (§5), an Intent-aware Capsule Graph Attention Network (§6) and a Cluster-based Historical Attention Mechanism (§7). All important symbols are marked in Fig. 4 for the easy check.

- **Message-Passing-based Routine Extraction (Section 5).** Noisy behaviors in the routines can cause the model to learn false correlations between behaviors. Therefore, we first perform routine extraction from UDI sequences by message passing and then apply the contrastive loss $\mathcal{L}_{CL}$ on behavior embedding $\mathbf{h}$ to

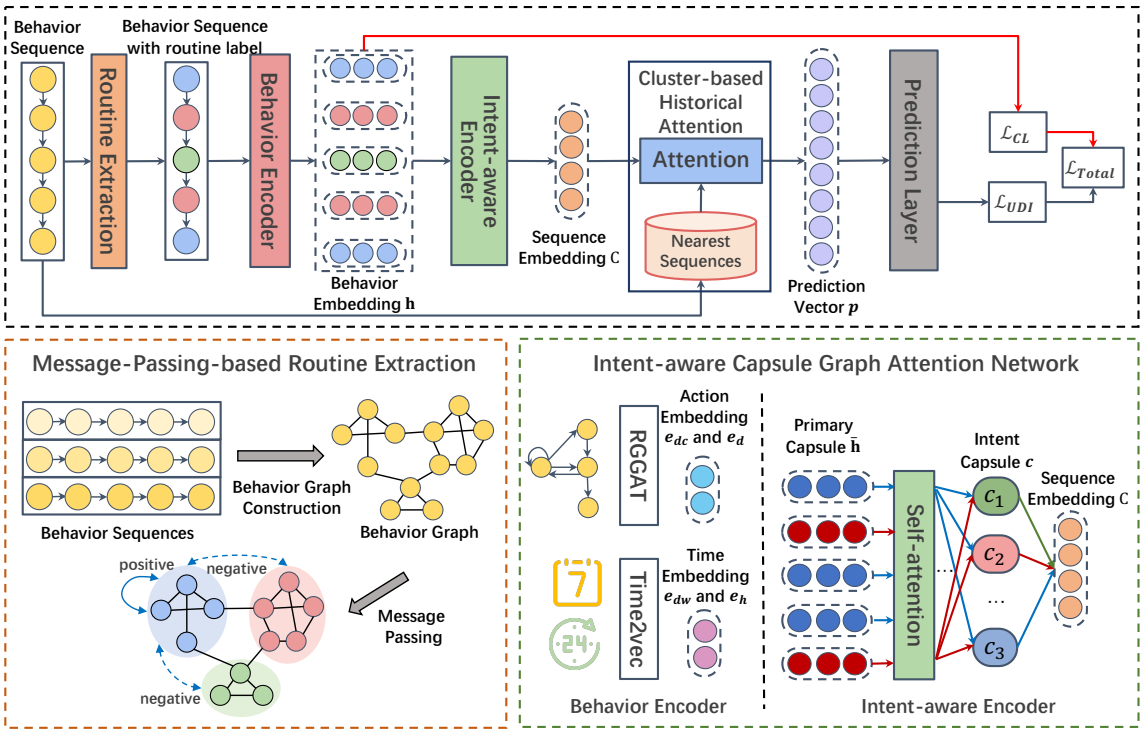


Fig. 4. Overview of SmartUDI. The blue, red, and green nodes in the MPRE part represent behaviors with different routines. The dark blue and dark red nodes in intent-aware encoder indicate primary capsules with different intents.

minimize the difference between behaviors within the same routine (positive samples) and maximize the difference between the behaviors derived from different routines (negative samples).

- **Intent-aware Capsule Graph Attention Network (Section 6).** For encoding multiple intents while considering the complex transitions between devices, the behavior sequence data is fed into **behavior encoder** consisting of Relational Gated Graph Attention Network (RGGAT) and Time2Vec [15] to learn behavior representation $\mathbf{h}$, then $\mathbf{h}$ are input to the **intent-aware encoder** to extract multi-intent representations of users for the sequence representations C.
- **Cluster-based Historical Attention Mechanism (Section 7).** We propose to capture the multi-level periodicity from historical sequences. Specifically, k-means [12] is applied to cluster behavior sequences and find semantically nearest historical sequences. Then attention mechanism is applied to aggregate the current sequence and the semantically nearest historical sequences representation to get the prediction vector p for the final prediction.

The workflow of SmartUDI is summarized as follows. First, Routine Extraction classifies the routines of each behavior sequence. The behavior sequences with their corresponding routine labels are fed into behavior Encoder. Second, Behavior Encoder encodes the behavior into the behavior embedding $\mathbf{h}$. Intent-aware Encoder takes the $\mathbf{h}$ as input and outputs the sequence embedding C. Third, CHAM applies the attention mechanism between the current sequence embedding C and the nearest historical sequence embeddings, where the scope of historical sequences is determined by the clustering algorithm (Section 7.1)

The MPRE and the clustering algorithm (Section 7.1) in CHAM run offline and are executed before training. Other modules such as ICGAT and CHAM are trained in an end-to-end fashion by backpropagation. The loss function is described in Section 8. The trained optimizer and related parameters are described in Section 9.1.4. The training process of SmartUDI can be performed on the cloud. Once trained, SmartUDI can be seamlessly deployed on either cloud infrastructures of IoT device vendors or edge nodes, allowing for real-time UDI prediction.

5 MESSAGE-PASSING-BASED ROUTINE EXTRACTION

We define a series of behaviors that often occur simultaneously as a routine, which reflects a person's behavior habits. Behaviors in the same routine are highly correlated, while behaviors of different routines are less correlated, mining this correlation is beneficial for UDI prediction. As mentioned before, there are often some noise behaviors mixed in a routine, so we design a Message-Passing-based Routine Extraction (MPRE) Algorithm to extract routines from behavior sequences. MPRE consists of a behavior graph construction and a routine extraction algorithm. The Behavior Graph is constructed according to the co-occurrence of different behaviors. Because behaviors in a routine often appear together in a sequence, closely related behaviors (in the same community) in the behavior graph are more likely to belong to a routine, so we designed a routine extraction algorithm to detect the community in the behavior graph.

Behavior Graph Construction. We construct behavior graph from all behavior sequences. The behavior graph can be modeled as a weighted directed graph $G_b(V_b, E_b)$. Each node in the graph represents a device control. Each edge $(c_i, c_j) \in E_b$ indicates device controls c_i and c_j have co-occurred in the same behavior sequence, with the weight of the edge represents the number of times that co-occurrence of c_i and c_j happens. A larger edge weight indicates a closer relationship between the two device controls. Lines 1-8 in Algorithm 1 show the construction process of the graph $G_b(V_b, E_b)$. Line 11 assigns each edge a normalized weight calculated as the weight of the edge divided by the sum of the weights of all outgoing edges of the starting node of that edge. Lines 12-14 remove edges with lower weight and normalized weight, which helps to separate irrelevant behaviors on the behavior graph. In line 12, we set $w\_threshold$ as 3 and $nw\_threshold$ as 0.1.

Through behavior graph construction, we transform the routine extraction problem into the community detection problem [16]. Communities in a graph are the groups of nodes, which are more highly connected to each other than to the rest of the nodes in the graph. Thus, extracting routines from behavior sequences is equivalent to identifying communities from the behavior graph.

Message Passing. The routine of a node is not only related to its own features, but also related to the features of other nodes in the behavior graph. Therefore, we perform message passing on the graph to realize node information exchange. Before message passing, lines 16-24 use the maximal clique [1] algorithm [37] to initialize the routine that the node belongs to, because each node in a maximal clique is connected to each other, and these device controls are likely to belong to the same routine. In line 17, K represents the number of maximal cliques in the behavior graph and $clique[k]$ denotes the set of nodes in k-th clique. Lines 25-34 update the routines of different nodes through message passing. For each node, the weighted messages of its neighbor nodes are accumulated, and then the updated routine is obtained by averaging its own routine and messages. After message passing, we normalize the routine feature of each node and assign routine labels to nodes based on the routine feature, as presented in lines 36-40. $routine\_feature[node][k]$ indicates the probability that $node$ belongs to routine k. Line 39 assigns the $node$ to the routine k with the highest probability.

Through routine extraction, related behaviors are grouped together and unrelated behaviors are separated. Based on the results of the routines, we can construct positive and negative samples which are used for contrastive learning (section 8). If two behaviors are from the same routine, they are treated as a positive pair. If they are

[1] A clique is a complete subgraph of a graph, that is, all nodes in a clique are connected to each other. A clique is called a maximal clique of a graph if it is not contained by any other clique. Here we use the Bron-Kerbosh algorithm to solve the maximum clique in the behavior graph.

Algorithm 1 Message-Passing-based Routine Extraction Algorithm.

Input: behavior sequences $\mathcal{S}$, routine number K, iteration number of message passing T
Output: routine class *routine*

```
// graph construction
for s ∈ S do
    for i = 1, ...len(s.device_controls) do
        for j = 1, ...len(s.device_controls) do
            edges[s.device_controls[i]][s.device_controls[j]] += 1;
        end for
    end for
end for
// remove the edge with small weight
for edge ∈ edges do
    edge_weight[i][j] = edges[i][j]/sum(edges[i]);
    if edges[i][j] >w_threshold && edge_weight[i][j] > nw_threshold then
        graph.add_edge(i,j,edge_weight[i][j]);
    end if
end for
// initialize the routine by clique
clique, K = get_maximal_clique(graph)
for node ∈ graph.nodes do
    for k = 1, ...K do
        if node ∈ clique[k] then
            routine_feature[node][k] = 1;
        end if
    end for
end for
// message passing
for iter = 1...T do
    for node ∈ graph.nodes do
        message = 0;
        for neighbor ∈ node.neighbors do
            message += routine_feature[neighbor]*edge_weight[node][neighbor];
        end for
        routine_feature[node] = (message+routine_feature[node])/2;
    end for
end for
// calculate the label of each node
for node ∈ graph.nodes do
    total = sum(routine_feature[node])
    routine_feature[node] = routine_feature[node] / total;
    routine[node] = argmax_{k∈{1...K}}(routine_feature[node][k])
end for
return routine
```

from different routines, they are treated as a negative pair. As shown in Fig. 4 (bottom left), for the behaviors in the blue area, the behaviors inside the blue area are the positive samples, and the behaviors in the red and green areas are the negative samples.

6 INTENT-AWARE CAPSULE GRAPH ATTENTION NETWORK

The Intent-aware Capsule Graph Attention Network mainly includes a behavior encoder and an intent-aware encoder. The behavior encoder (§6.1) encodes device control with device and time to fully account for contextual information. The intent-aware encoder (§6.2) views the behavior representations as a primary capsule and learns multi-intents representations by dynamic routing.

6.1 Behavior Encoder

Because human behaviors consist of time, device and device control, to fully account for contextual information, we design different ways to learn the embeddings of time, device and device control respectively.

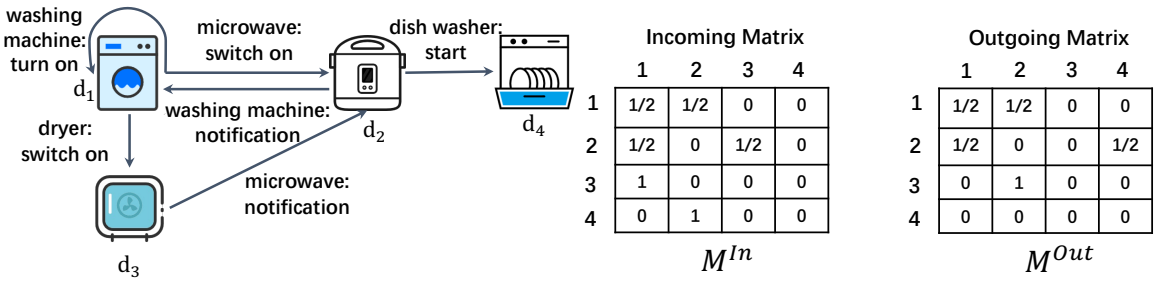
6.1.1 Time Embedding. Due to the continuity of timestamps, it is impractical to learn temporal representations directly from timestamps. We express time as hour of the day and day of the week, as they are shown to affect users' device controls [13]. We leverage Time2Vec [15] to learn time embedding because 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 $e_{dw}, e_h \in \mathbb{R}^L$ denote the embeddings of day of the week and hour of the day which are obtained by Time2Vec, respectively.

6.1.2 Device and Device Control Embeddings. When learning device and device control embeddings, we need to consider complex heterogeneous transitions between different devices and device controls. [38] 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 [3], 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.

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. 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 Fig. 5(a) and the relevant incoming and outgoing matrices are shown in the Fig. 5(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.



(a) An example of relational sequence graph.

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

Fig. 5. Construction of RSGraph.

Device Embedding Update on RS-Graph. Next, we describe how to learn representations for each device and device control with Relational Gated Graph Attention Neural Networks (RGGAT).

To learn representations for devices, we first encode each device $d \in \mathcal{D}$ into a low-dimensional latent space through an embedding layer. 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:

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

where $\mathbf{W}$ is the shared weight matrix, and Attention is the attention mechanism [36], i.e., a single-layer feedforward neural network and the LeakyReLU nonlinear activation function. 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 e_{dc} 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.

6.1.3 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:

$$\tilde{\mathbf{h}} = [e_{dw}, e_h, e_d, e_{dc}], \tag{11}$$

Since our model contains no recurrence and no convolution, we must inject some information about the relative or absolute position of the behaviors in the sequence so that the model can make use of the order of the sequence. To identify the position of the input variable, following [35], 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 $\tilde{\mathbf{h}}$.

6.2 Intent-aware Encoder

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.

6.2.1 Self-attention Layer. We employ transformer encoder [35] 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 [35].

6.2.2 Capsule Networks. We utilize Capsule Networks [29] (CapsNets) to extract multiple intents of a behavior sequence. We can naturally treat intents as capsules to learn the probabilities of various intents, since the capsule network learns the probability that a capsule's entity exists. 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 [29]. 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 SmartUDI, 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 Fig. 4 (bottom 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 2. Then a non-linear "squashing"

function [29] 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 $len(c)$ denotes the dimension of primary capsules. 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 2 SmartUDI Dynamic Routing.

Input: primary capsules $\overline{\mathbf{h}}_i$, iteration times T, number of intent capsules K
Output: intent capules $\left\{\mathbf{c}_j, j = 1, \ldots, K\right\}$

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: $\mathbf{w}_i = \text{softmax}\left(\mathbf{b}_i\right)$.
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: $\mathbf{c}_j = \text{squash}\left(\mathbf{s}_j\right)$.
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** $\left\{\mathbf{c}_j, j = 1, \ldots, K\right\}$

7 CLUSTER-BASE HISTORICAL ATTENTION MECHANISM

[9, 32] show the multi-level nature of human behavior periodicity, a static representation of user behavior sequences fails to capture the periodicity and evolution of user behavior. Human behavior is dynamic in nature. The multi-level periodicity (day-level, week-level, month-level, and even some irregular periodicity) may exist at the same time in human behaviors, we cannot explicitly represent the periodicity times. Nevertheless, the existence of periodicity also means some human historical behaviors may reappear in a predictable way. By capturing the correlation between the current behavior sequence and the historical behavior sequences, such recurrence feature can be well excavated for predicting the next behavior, which essentially reflects the multi-level periodicity of human behavior. The attention mechanism [35] is often used to measure the correlation between two instances. DeepMove [9] mines the multi-level periodicity of human mobility by applying the attention mechanism between the current sequence and all historical sequences. However, considering all historical sequences not only increase the training overhead as the sequence data increases, but also introduces noise since not all historical sequences are related to the current sequence. Therefore, we design a Cluster-based Historical Attention Mechanism (CHAM). CHAM first clusters behavior sequences according to their semantic similarities and then applies historical attention mechanism within the scopes of clusters. By considering the historical sequences semantically nearest to the current sequence, CHAM not only reduces the training overhead, but also filters the irrelevant sequences, thereby improving the prediction performance.

7.1 Behavior Sequences Clustering

Behavioral Sequence Clustering needs to be completed before training, so as to ensure that the model can obtain similar historical sequences according to the clustering results during training. To effectively cluster behavior sequences, we first need to evaluate the semantic similarity between behavior sequences which jointly considers the device, device control and time dimensions.

Edit distance [30] is widely used to measure the similarity between two sequences. It is calculated by the minimum number of editing operations required to convert one sequence to another. A smaller distance means more similar sequences. However, edit distance is not feasible to measure the similarity between behavior sequences in our scenarios due to the following two reasons. First, edit distance does not incorporate device control semantic information. For example, suppose device control sequence A is "turn on light -> turn on outlet -> turn off light", device control sequence B is "turn on light -> adjust outlet mode -> turn off light", and device control sequence C is "turn on light -> close curtain -> turn off light". The edit distance between the three sequences is 1, but the degree of difference between A and B is lower than that between A and C, because the second behaviors of A and B are both to control on the outlet, while the second behavior of C is to control the curtain. Second, edit distance does not consider the time when a behavior occurs, which is also of great value for evaluating similarity.

7.1.1 Device and Device Control Similarity. Node2vec [11] can learn low-dimensional representations containing semantic information for nodes in a graph by using random walks through a graph starting at a target node. We have constructed RS-Graphs for devices and device controls in section 6.1, so we perform node2vec [11] on RS-Graphs to learn the representations of device and device control.

7.1.2 Time Similarity. Because behaviors are cyclical, using the absolute time to measure time similarity between behaviors is inappropriate. For example, a user may open the curtains at 8:00 a.m. on both Tuesday and Wednesday. Though the absolute time dissimilarity is one day, in fact, there is no difference between the two behaviors because they both occur at 8:00 a.m.. Therefore, we express time as the hour of the day. Then, the time t is converted to the radian of a unit circle in the coordinates centered on (0, 0), i.e., $[0, 24) \rightarrow [0, 2\pi)$. The hour time is represented by the coordinate of a point in the unit circle based on the radian θ [21] following:

$$H(t) = (\cos\theta, \sin\theta), \theta = 2\pi\left(\frac{t}{24}\right). \tag{22}$$

Let $F(d)$ and $F(dc)$ denote the device and the device control representations learned by node2vec [11], respectively. The similarity between two behavior sequences s_p and s_q is computed as:

$$d(s_p, s_q) = \sum_{pi=1,qi=1}^{len(s)} (F(d_{pi})^T F(d_{qi}) + F(dc_{pi})^T F(dc_{qi}) + H(t_{pi})^T H(t_{qi})), \tag{23}$$

where $len(s)$ is the length of sequence s_p and s_q, pi and qi are the pi-th and qi-th item in the behavior sequence s_p and s_q, respectively.

Finally, we use the k-means algorithm [12] to cluster behavioral sequences, where the similarity of the sequences is measured by Eq.23.

7.2 History Attention Mechanism

Suppose the current sequence is s_t, the nearest history sequences $\{s_i \in Cluster(s_t)\}$ are obtained from the cluster results. 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 nearest historical behavior sequences. The historical attention mechanism is formulated as follows:

$$\alpha_i = \frac{\exp(\beta_i)}{\sum_{j=1}^{t-1}\exp(\beta_j)}, \quad \beta_i = \tanh(\mathbf{C}_t \mathbf{W}_H \mathbf{C}_i), \tag{24}$$

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 in the nearest sequences 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:

$$p = \text{Concat}\left(\mathbf{C}_t, \sum_{i=1}^{t-1}\alpha_i \mathbf{C}_i\right), \tag{25}$$

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

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_p p), \tag{26}$$

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.

8 MULTI-TASK TRAINING

Contrastive learning as a self-supervised learning method has been applied to make the representation vectors of "similar" samples close and those of "dissimilar" samples alienated [5]. To leverage the self-supervised signals derived from the routine extraction part (section 5) to enhance the performance of UDI prediction, we adopt a multi-task strategy where the main UDI prediction task and the contrastive learning task are jointly optimized.

Contrastive Learning Task Loss Function. Based on the routine, a contrastive loss function is used to distinguish whether two behaviors originate from the same routine. The contrastive loss can lean to minimize the difference between behaviors of the same routine and maximize the difference between the behaviors derived from different routines. Through contrastive learning, the correlation of behaviors in the same routine can be strengthened, and the correlation of behaviors not in the same routine will be reduced, thereby solving the problem of false correlation. We first define the similarity between two behavior representations $\mathbf{h}_p$ and $\mathbf{h}_q$ as:

$$\text{sim}(\mathbf{h}_p, \mathbf{h}_q) = \mathbf{h}_p^\top \mathbf{h}_q / \|\mathbf{h}_p\| \|\mathbf{h}_q\|. \tag{27}$$

Then, the contrastive loss for $\mathbf{h}_i$ in behavior sequence s is defined as:

$$\mathcal{L}(\mathbf{h}_i) = -\log \frac{\sum_{\mathbf{h}_j \in pos(\mathbf{h}_i), \mathbf{h}_j \neq \mathbf{h}_i} \exp(\text{sim}(\mathbf{h}_i, \mathbf{h}_j)/\tau)}{\sum_{\mathbf{h}_k \in neg(\mathbf{h}_i), \mathbf{h}_k \neq \mathbf{h}_i} \exp(\text{sim}(\mathbf{h}_i, \mathbf{h}_k)/\tau)}, \tag{28}$$

where $pos(\mathbf{h}_i)$ is set of positive samples for $\mathbf{h_i}$, $neg(\mathbf{h}_i)$ is the set of negative samples for $\mathbf{h_i}$, and τ is a temperature parameter to control the sensitivity of the cosine similarity. Finally, the contrastive loss function can be defined as:

$$\mathcal{L}_{CL} = \frac{1}{\|\mathcal{S}\|} \sum_{s \in S} \sum_{\mathbf{h}_i \in s} \mathcal{L}(\mathbf{h}_i), \tag{29}$$

where $\|\mathcal{S}\|$ is the number of behavior sequences.

UDI Prediction Task Loss Function. We define the cross-entropy loss function for user device interaction prediction problem as follows:

$$\mathcal{L}_{UDI}(X, \mathrm{Y}) = -\frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathbf{y}_s \log \hat{\mathbf{y}}_s, \tag{30}$$

where $\hat{\mathbf{y}}_\mathbf{s}$ is the predicted probabilities of the next device control of the behavior sequence s and $\mathbf{y}$ is the one-hot vector of the ground-truth label.

The total loss is a linear weighted sum:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{UDI}} + \lambda \mathcal{L}_{\text{CL}}, \tag{31}$$

where λ is a balance hyperparameter.

9 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, can SmartUDI predict user device interaction more accuractely?
- RQ2. **Ablation study.** How does each main component of SmartUDI affects the performance of UDI prediction?
- RQ3. **Parameter study.** How do key parameters affect the performance of SmartUDI?
- RQ4. **Interpretability study.** Can SmartUDI give a reasonable explanation for the prediction results?
- RQ5. **Embedding space analysis.** Does SmartUDI successfully learn useful embedding vectors of behaviors and correct correlations between behaviors?

9.1 Experimental Setup

9.1.1 Datasets. We evaluate model performance using four real-world smart home datasets, three (FR/SP/US) from public datasets[2] and one anonymous dataset (AN) collected by ourselves. The datasets description is shown in Table 1, the testbed and collection process of AN datasets are shown in Appendix B. In the datasets, all behavior sequences are arranged in chronological order. All datasets are split into training, validation and testing sets with a ratio of 7:1:2 according to the timestamps of the behaviors. The divided datasets still maintain time order (from early to late). That is, all behaviors in the validation set occur after those in the training set and all behaviors in the test set occur after those in both the training and validation sets. We create sequential instances with a window of length 10. The first nine behaviors of the window are input of SmartUDI for predicting the next behavior, i.e., the 10th behavior. Time features of a behavior is a combination of the day of week and hour based on the behavior's timestamp. The hour is one of the 8 time ranges of 3 hours length: 0-3, 3-6, 6-9, 9-12, 12-15, 15-18, 18-21, and 21-24. It is worth noting that we divide the behavior into fixed time intervals (i.e., 3h), so that the time span of a user behavior sequence depends on the total time span of the behaviors in the sequence. We label user behaviors with intents based on device attributes. For example, an oven belongs to the cooking intent, and a TV belongs to the entertainment intent. Information of all intents is included in Appendix C.

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

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

9.1.2 Baselines. Several methods in existing studies about UDI prediction in smart home and sequential user behavior prediction are employed as baseline algorithms:

- **HMM** [2] builds a transition matrix between different device controls to capture the first order transition probabilities for UDI prediction.
- **FPMC** [28] combines Markov chain with matrix factorization to capture both sequential patterns and user preferences for UDI prediction.
- **LSTM** [34] captures the long-term sequential influence for UDI prediction.
- **CARNN** [23] considers context-dependent features by using context-specific transition matrix in RNN cell for a sequential recommendation.
- **Caser** [33] employs CNN in both time-axis and feature-axis to capture temporal dynamics for UDI prediction.
- **DeepMove** [9] captures both long and short-term user behavior patterns by the enhanced version of RNN with a history attention mechanism.
- **SIAR** [27] applies Stacked Recurrent Neural Networks that model the dynamics of contexts and temporal gaps for context-aware UDI prediction.
- **SR-GNN** [38] applies gated graph neural network to generate latent vectors of items and then represents each session through an attention network for user behavior prediction.
- **SASRec** [14] uses time-growing directional transformer encoder to consider sequential patterns of user actions for action prediction.
- **SmartSense** [13] applies query transformer for smart home action recommendation.
- **DeepUDI** [39] applies graph neural network, transformer and attention mechanism to predict user behavior in smart home scenarios.

9.1.3 Evaluation Metrics. Top-k accuracy (Acc@k) and Macro-F1 are adopted as evaluation metrics for UDI prediction. Specifically, let $s \in S$ denote the set of behavior sequences and $P_K(s)$ denote the set of K device controls with the largest probabilities that the model predicts for the behavior sequence s, the Acc@K can be formulated as:

$$\text{Acc@K} = \frac{\mid \{s \in S : p(s) \in P_K(s)\} \mid}{|S|}, \tag{32}$$

where $p(s)$ denotes the ground truth of the next device controls. As UDI prediction is a multi-class classification problem and the class distribution is uneven (as shown in Appendix A), Macro-F1 is a more generic metric to the evaluation of model performance on a dataset with an uneven class distribution:

$$\text{Macro-F1} = \frac{\sum_c \text{F1}_c}{|C|}, \tag{33}$$

where F1_c denotes the F1 score of class c (device control), and the F1 score is the harmonic mean of the precision and recall.

9.1.4 Implementation. All models are implemented by PyTorch [26] and run on a graphic card of GeForce RTX 3090 Ti. All models are trained with Adam optimizer [18] with a learning rate of 0.001. The iteration number T in Algorithm.1 is set to 2 and the balance hyperparameter λ in Eq.31 is set to 0.5. SmartUDI is trained by multi-task training strategy in Section 8. During training, we monitor Acc@1 and stop training if there is no performance improvement in 10 steps. We observe the time cost of SmartUDI for next behavior prediction is 11.8ms, which means that SmartUDI is capable of real-time UDI prediction.

Table 2. Performance comparison on four real-world datasets.

Dataset	Metric	HMM	FPMC	LSTM	CARNN	Caser	DeepMove	SIAR	SRGNN	SASRec	SmartSense	DeepUDI	SmartUDI
AN	Acc@1	0.6099	0.6557	0.7062	0.7026	0.7054	0.7116	0.7238	0.9245	0.9325	0.9407	0.9784	**0.9805**
	Acc@3	0.7501	0.7959	0.7843	0.8302	0.8569	0.9272	0.9354	0.9864	0.9665	0.9731	0.9865	**0.9874**
	Acc@5	0.7714	0.7902	0.8328	0.9003	0.9273	0.9542	0.9645	0.9872	0.9765	0.9838	0.9882	**0.9892**
	Macro-F1	0.2439	0.2845	0.3759	0.4159	0.4467	0.5027	0.5259	0.7368	0.7432	0.7519	0.7997	**0.8966**
FR	Acc@1	0.6536	0.6814	0.6962	0.7893	0.7742	0.7762	0.7796	0.7819	0.7821	0.7923	0.8144	**0.8145**
	Acc@3	0.7813	0.8271	0.8011	0.9148	0.9201	0.9221	0.9120	0.9197	0.9204	0.9232	0.9237	**0.9238**
	Acc@5	0.8242	0.8508	0.8565	0.9425	0.9414	0.9446	0.9420	0.9435	0.9362	0.9379	0.9511	**0.9512**
	Macro-F1	0.1127	0.1279	0.1302	0.2102	0.2158	0.2288	0.2312	0.2482	0.2473	0.2603	0.3425	**0.3837**
SP	Acc@1	0.6315	0.6964	0.7517	0.7853	0.7721	0.7756	0.7802	0.7815	0.7821	0.7921	0.7923	**0.7930**
	Acc@3	0.7863	0.7916	0.8864	0.8915	0.9045	0.9125	0.9217	0.9303	0.9321	0.9342	0.9375	**0.9427**
	Acc@5	0.8361	0.8605	0.9346	0.9117	0.9273	0.9521	0.9597	0.9603	0.9560	0.9511	0.9642	**0.9671**
	Macro-F1	0.1382	0.1586	0.1756	0.1745	0.1927	0.2159	0.2176	0.2239	0.2254	0.2244	0.3112	**0.3328**
US	Acc@1	0.3327	0.3543	0.4286	0.5212	0.5378	0.5527	0.5633	0.5784	0.5826	0.5935	0.6056	**0.6321**
	Acc@3	0.6881	0.6992	0.8209	0.8577	0.8632	0.8844	0.8902	0.8955	0.8972	0.9056	0.9123	**0.9058**
	Acc@5	0.7258	0.7712	0.8929	0.9135	0.9266	0.9418	0.9432	0.9463	0.9320	0.9489	0.9521	**0.9538**
	Macro-F1	0.1069	0.1123	0.1265	0.1396	0.1576	0.2388	0.2397	0.2431	0.2433	0.2451	0.3538	**0.3742**

9.2 Performance Comparison (RQ1)

We use grid search to adjust the parameters (§9.4) of SmartUDI and select the optimal results. The results are shown in Table 2. For each row, the best performance and the second best performance are highlighted by bold and underline, respectively. We can make the following observations: 1) The proposed SmartUDI scheme outperforms all baselines; 2) the traditional models, HMM and FPMC, show the worst performance; 3) the CNN-based and RNN-based models outperform the traditional models; 4) SR-GNN outperforms RNN-based models; 5) Transformer-based models SASRec, SmartSense and DeepUDI achieve better performance than all other baselines. Nevertheless, their performance is still inferior to that of SmartUDI.

9.3 Ablation Study (RQ2)

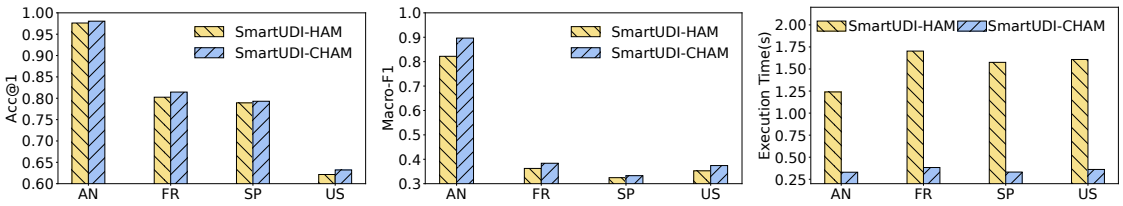
Table 3. Ablation study results on four datasets.

Model	AN		FR		SP		US	
	Acc@1	Macro-F1	Acc@1	Macro-F1	Acc@1	Macro-F1	Acc@1	Macro-F1
SmartUDI(w/o MPRE)	0.9622	0.7689	0.7871	0.3093	0.7832	0.3072	0.6028	0.3375
SmartUDI(w/o ICGAT)	0.9568	0.7354	0.7773	0.3199	0.7763	0.2846	0.5872	0.3252
SmartUDI(w/o CHAM)	0.9757	0.8018	0.7929	0.3458	0.7881	0.3235	0.6125	0.3486
SmartUDI(w/o ALL)	0.9137	0.6934	0.7578	0.2511	0.7685	0.2482	0.5736	0.2476
SmartUDI	**0.9805**	**0.8966**	**0.8145**	**0.3837**	**0.7930**	**0.3328**	**0.6321**	**0.3742**

To verify the contribution of each of the main components of SmartUDI, i.e., Message-Passing-based Routine Extraction (MPRE), Intent-aware Capsule Graph Attention Network (ICGAT) and Cluster-based Historical Attention Mechanism (CHAM), to the final prediction results, we conduct ablation experiments with the optimal parameters unchanged. SmartUDI(w/o MPRE) denotes SmartUDI without routine extraction and contrastive learning process. SmartUDI(w/o ICGAT) denotes SmartUDI without ICGAT, which, instead of behavior encoder

and intent-aware encoder, uses a simple embedding layer (i.e., a fully connected layer). SmartUDI(w/o CHAM) denotes SmartUDI without CHAM. SmartUDI(w/o ALL) denotes SmartUDI with none of the three components. SmartUDI is the full scheme with all the three components. Table 3 shows that SmartUDI outperforms all others on all the four datasets, while SmartUDI(w/o ALL) shows the worst Acc@1 and Macro-F1. In summary, each of the three components of SmartUDI is contributive for UDI prediction.

Furthermore, we conduct the abalation study on CHAM, and the results are shown in Fig. 6. SmartUDI-HAM applies attention to all historical behavior sequences and SmartUDI-CHAM cluster behavior sequences first and then applies attention to semantically nearest historical sequences. Fig. 6(a) and Fig. 6(b) show that the Acc@1 and Macro-F1 of SmartUDI-CHAM are higher than those of SmartUDI-HAM, because semantically nearest historical sequences are more helpful for prediction. Then, we set the batch size to 512 to test the execution time of the model. Fig. 6(c) shows that the execution time per batch of SmartUDI-CHAM is lower than that of SmartUDI-HAM, which means that SmartUDI-CHAM has higher execution efficiency.

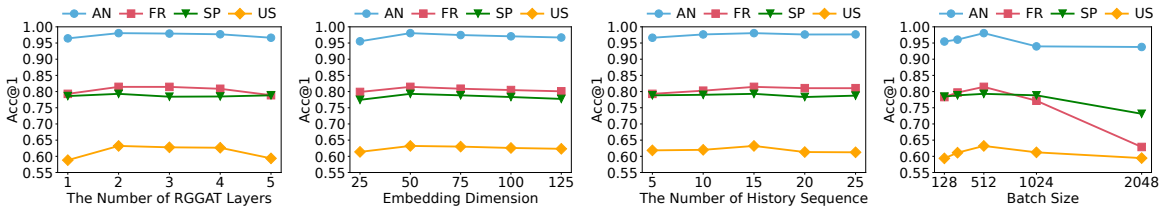


(a) Acc@1. (b) Macro-F1. (c) Execution time per batch.

Fig. 6. Abalation study on Cluster-based Historical Attention Mechanism.

9.4 Influence of Hyper-parameters (RQ3)

We conduct experiments to explore the effects of the number of RGGAT layers, embedding dimension, number of historical behavior sequence and batch size by keeping other optimal parameters unchanged.



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

Fig. 7. The influence of hyper-parameters on Acc@1.

9.4.1 Number of Layers in RGGAT. Fig. 7(a) shows the performance of SmartUDI with different numbers of RGGAT layers. We find that when the number of RGGAT layers increases, Acc@1 first increases and then decreases, reaching the optimal value at 2 layers. This is because too few layers can lead to under-fitting, while too many layers can lead to over-smoothing [4].

9.4.2 Embedding Dimension. As shown in Fig. 7(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 gradually increases. A larger embedding size does not necessarily lead to better performance because of over-fitting. Therefore, we choose the embedding size to be 50 to achieve the best performance.

9.4.3 Number of History Behavior Sequences. When performing CHAM, we consider a fixed number of historical sequences which are most similar to the current sequence in the same cluster. A larger number of historical sequences allows the model to consider more historical information. However, too much historical information may introduce more uncertainty and is no longer beneficial for performance. Fig. 7(c) shows that 15 historical sequences enable SmartUDI to achieve the optimal performance.

9.4.4 Batch Size. Fig. 7(d) shows the influences of batch size. As the batch size increases, Acc@1 increases. When the batch size exceeds 512, the increase in batch size leads to a decrease in performance since larger batch size hurts the generalization ability of the model.

9.5 Case Study (RQ4)

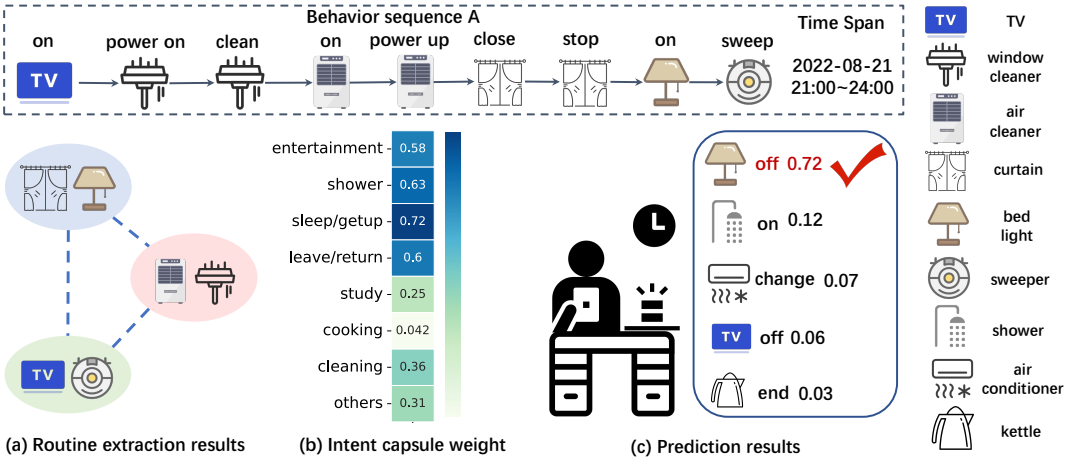


Fig. 8. (a) Routine extraction results, (b) capsule weight and (c) top 5 prediction results of the example.

To verify the interpretability of SmartUDI, we choose a behavior sequence A from the test set of the AN dataset and visualize the routine extraction results of Message-Passing-based Routine Extraction (MPRE), multi-intent weights of Intent-aware Capsule Graph Attention Network (ICGAT) and history attention score of Cluster-based Historical Attention Mechanism (CHAM). The results are shown in Fig. 8 and Fig. 9, we can make the following three observations.

First, as shown Fig. 8(a), MPRE can dig out potential device correlations from behavior sequences by routine extraction, which reflect the user's behavior habits. The routine extraction results show that there are three routines represented as different areas with different colors in the behavior sequence A. Devices in the same routine have a high correlation. The correlation between curtains and bedlight is high because both are operated right before bedtime and after getting up. The high correlation between the sweeper and TV shows that the user tends to do some entertainment activities while doing housework activities. The air cleaner and window cleaner are highly relevant because they are both cleaning devices.

Second, as shown Fig. 8(b), ICGAT can effectively mine users' multiple intents and learn the importance of different intents for final prediction. We visualize the intent weight $\mathbf{W}_C$ in Eq. 21. 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 explain 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 as shown in Fig. 8(c). The reason why the predicted probabilities of "*shower:on*" and "*air conditioner:change*" are also 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 has already turned on the TV and needs to turn it off sometime before bed.

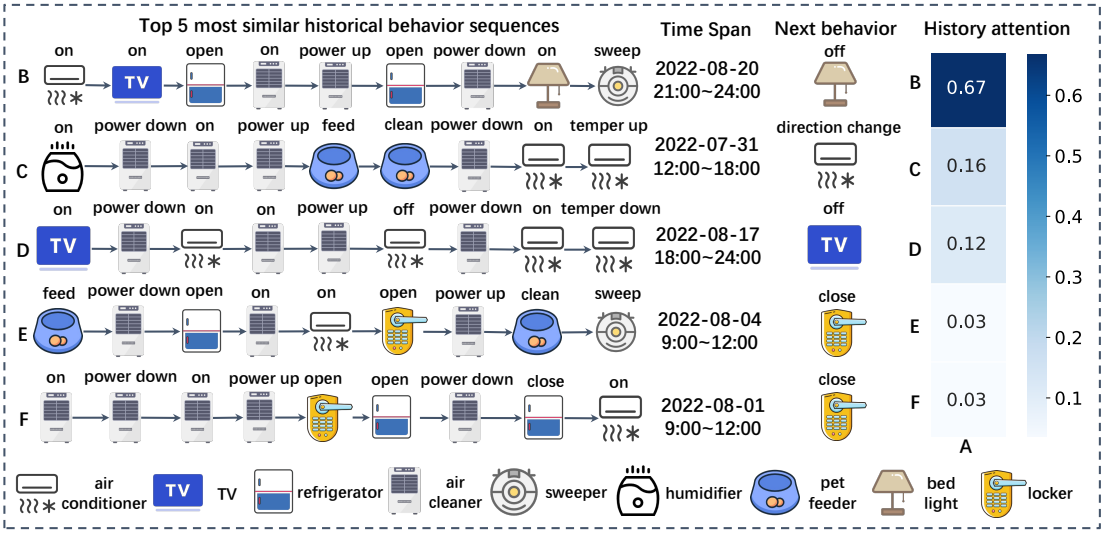
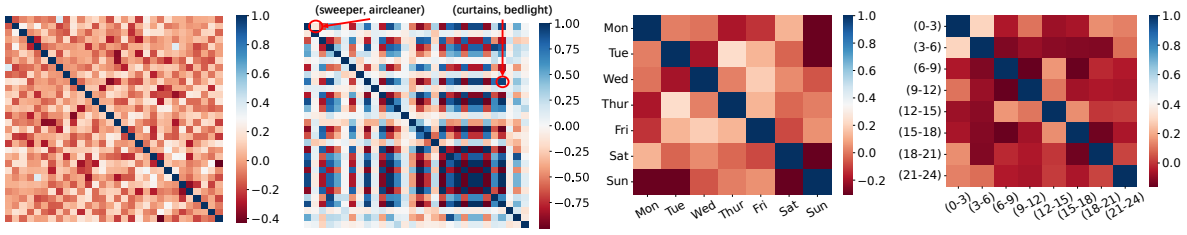


Fig. 9. Top 5 most similar historical sequences, time span, next behavior and historical attention score.

Third, as shown Fig. 9, CHAM can leverage most similar historical behavior sequences to model multi-level periodicity for assisting the prediction. The top five most similar historical behavior sequences (B, C, D, E, F) and historical attention scores between the behavior sequence A and historical behavior sequences are shown in Fig. 9. From B to F, the similarity is gradually decreasing, correspondingly, the attention score also gradually decreases. The similarity and attention score between B and A are the highest, so when applying CHAM, historical sequence B plays the most important role in predicting A's next behavior. The next behavior of sequence B is "bedlight:off", so there is a high probability that the next behavior of sequence A is "bedlight:off". As shown in Fig. 9, We find that behavior sequence A occurred between 21:00 and 24:00 of 2022-8-21 and behavior sequence B occurred between 21:00 and 24:00 of 2022-8-20, so we can speculate that A and B show day-level periodicity.

9.6 Embedding Space Analysis (RQ5)

We analyze whether the model effectively learns the relationship between behaviors by observing the embedding space of the device and time.



(a) Device embedding w/o CL. (b) Device embedding with CL. (c) Week embedding. (d) Hour embedding.

Fig. 10. The device embedding similarity matrix and time embedding similarity matrix.

9.6.1 Device Embedding. We visualize the similarity between device embeddings before and after applying contrastive learning. The results are shown in Fig. 10(a) (device embedding without contrastive learning) and Fig. 10(b) (device embedding with contrastive learning). As shown in Fig. 10(a), before contrastive learning, the similarity between almost all devices is very low, because there is noise behavior in the behavior sequence, and it is difficult for the model to mine the correlation between different devices. In Fig 10(b), we can see that the correlation between “curtain" and “bedlight" is relatively high, while the correlation between “sweeper" and “aircleaner" is relatively low. Combining the results of Fig. 8, we can find that the “curtains" and “bedlight" are in the same routine, while the “sweeper" and “aircleaner" are in different routines. This shows that after applying contrastive learning on different routines, embeddings of devices in same routine get closer while embeddings of devices from different routines become more distant. This is because the contrastive learning narrows embeddings among devices from the same routines but alienates embeddings among devices from different routines.

9.6.2 Time Embedding. We visualize the day of week embedding and hour of day embedding to verify whether the model captures the time information well. The results are shown in Fig. 10(c) and Fig. 10(d). In general, the similarity between adjacent times is relatively high. For example, the similarity between 0-3 hours and 3-6 hours is high and the similarity between Thursday and Friday is high. However, human behavior is periodic, so the similarity between some non-adjacent times can also be high, e.g., Tuesday and Thursday, 6-9 hours and 12-15 hours.

10 DISCUSSION

While SmartUDI achieves excellent performance in the UDI prediction tasks, it has some room for further improvement. First, SmartUDI does not consider the impacts of environmental factors on human behaviors in smart homes, such as temperature, humidity, and brightness. If the temperature in the room is high, the user is likely to turn down the temperature of the air conditioner. If it is dark indoor (e.g., a cloudy day), the user may turn on lights even in the morning. In future work, we will consider collecting environmental factors from smart home sensors (such as temperature, humidity, light sensors), and feed these environmental features to the model when making UDI prediction to account for their impacts. Second, SmartUDI considers the behavior of the current user when making UDI predictions. There may be other users in the smart home, and the behaviors of different users may affect each other. In future work, we will consider the influence between behaviors from different users by exploiting the correlation between the current user’s behaviors and other users’ behaviors for better UDI prediction in multi-user scenarios.

11 CONCLUSION

In this paper, we propose SmartUDI, a novel user device interaction prediction framework, which achieves accurate UDI prediction by considering routine, intent, and multi-level periodicity of user behaviors. First, to extract routine behaviors from behavior sequences with noise, we propose Message-Passing-based Routine Extraction (MPRE) Algorithm and apply the contrastive loss to minimize the difference between behaviors within the same routine and maximize the difference between the behaviors derived from different routines. Second, we propose an Intent-aware Capsule Graph Attention network (ICGAT), which consists of a relational gated graph attention network and a capsule network to learn user multi-intent representations. Third, we propose a Cluster-based Historical Attention Mechanism (CHAM) to learn the multi-level periodicity of user behaviors efficiently from semantically nearest history sequences. We build a testbed and conduct comprehensive experiments on four real-world datasets. The results demonstrate that 1) SmartUDI outperforms all existing traditional, CNN-based, RNN-based, GNN-based and Transformer-based models; 2) our proposed three modules MPRE, ICGAT and CHAM are all contributive for UDI prediction; 3) the prediction results of SmartUDI can be well explained by analyzing the intents and historical sequences learned by the model; 4) by analyzing embedding space, we find that SmartUDI has learned both device-level correlation and time-level correlation very well.

ACKNOWLEDGMENTS

We would like to thank our AE and other anonymous reviewers for their constructive comments. We thank Yucheng Huang, Guanglin Duan and Xudong Zuo for volunteering. We also thank Lianjin Ye, Zhengxin Zhang, Wenxin Tang, Runjie Zhou, Xinchao Li, Zesheng Wang, Xiaobo Zhang and Liudi Shen for joining the discussion of this paper. This work is supported by the National Key Research and Development Program of China under grant No. 2022YFB3105000, the National Natural Science Foundation of China under grant No. 61972189, the Major Key Project of PCL under grant No. PCL2023AS5-1, 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] Miao Lin andf 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

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

[4] 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.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[6] Yegang Du, Yuto Lim, and Yasuo Tan. 2019. A Novel Human Activity Recognition and Prediction in Smart Home Based on Interaction. *Sensors* 19, 20 (2019), 4474. https://doi.org/10.3390/s19204474

[7] Sean R Eddy. 1996. Hidden markov models. *Current opinion in structural biology* 6, 3 (1996), 361–365.

[8] Araf Farayez, Mamun Bin Ibne Reaz, and Norhana Arsad. 2019. SPADE: Activity Prediction in Smart Homes Using Prefix Tree Based Context Generation. *IEEE Access* 7 (2019), 5492–5501. https://doi.org/10.1109/ACCESS.2018.2888923

[9] 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.

[10] Alex Graves and Alex Graves. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks* (2012), 37–45.

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.
[12] John A Hartigan and Manchek A Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)* 28, 1 (1979), 100–108.
[13] 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.
[14] 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.
[15] 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).
[16] Bisma S Khan and Muaz A Niazi. 2017. Network community detection: A review and visual survey. *arXiv preprint arXiv:1708.00977* (2017).
[17] Phil Kim and Phil Kim. 2017. Convolutional neural network. *MATLAB deep learning: with machine learning, neural networks and artificial intelligence* (2017), 121–147.
[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[19] Ruoyu Li, Qing Li, Yucheng Huang, Wenbin Zhang, Peican Zhu, and Yong Jiang. 2022. IoTEnsemble: Detection of Botnet Attacks on Internet of Things. In *European Symposium on Research in Computer Security (ESORICS)*. Springer, 569–588.
[20] Ruoyu Li, Qing Li, Jianer Zhou, and Yong Jiang. 2021. ADRIoT: an edge-assisted anomaly detection framework against IoT-based network attacks. *IEEE Internet of Things Journal (IoTJ)* 9, 13 (2021), 10576–10587.
[21] Chengwu Liao, Chao Chen, Suiming Guo, Zhu Wang, Yaxiao Liu, Ke Xu, and Daqing Zhang. 2022. Wheels know why you travel: Predicting trip purpose via a dual-attention graph embedding network. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)* 6, 1 (2022), 1–22.
[22] 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
[23] 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.
[24] 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/.
[25] 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
[26] 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).
[27] 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.
[28] 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.
[29] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. *Advances in Neural Information Processing Systems (NIPS)* 30 (2017).
[30] Klaus U Schulz and Stoyan Mihov. 2002. Fast string correction with Levenshtein automata. *International Journal on Document Analysis and Recognition* 5 (2002), 67–85.
[31] Amit Kumar Sikder, Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. 2019. Aegis: a context-aware security framework for smart home systems. In *Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019*, David Balenson (Ed.). ACM, 28–41. https://doi.org/10.1145/3359789.3359840
[32] 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.
[33] 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.
[34] 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

[35] 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).

[36] 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).

[37] Qinghua Wu and Jin-Kao Hao. 2015. A review on algorithms for maximum clique problems. *European Journal of Operational Research* 242, 3 (2015), 693–709.

[38] 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.

[39] 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 *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 1634–1642.

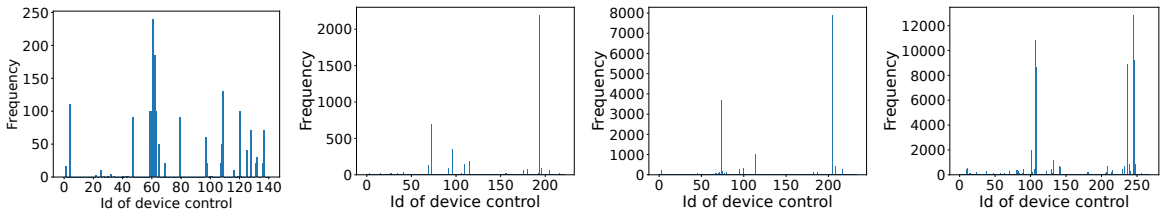
[40] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation.. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*,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

[41] 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

[42] Qingsong Zou, Qing Li, Ruoyu Li, Yucheng Huang, Gareth Tyson, Jingyu Xiao, and Yong Jiang. 2023. IoTBeholder: A Privacy Snooping Attack on User Habitual Behaviors from Smart Home Wi-Fi Traffic. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)* 7, 1 (2023), 1–26.

A DATA DISTRIBUTION

The class distribution of four datasets (AN,FR,SP,US) are shown in Fig. 11, we can find all the class distributions are uneven.



(a) The class distribution of AN. (b) The class distribution of FR. (c) The class distribution of SP. (d) The class distribution of US.

Fig. 11. The class distribution of four datasets.

B DESCRIPTION OF ANONYMOUS DATASET

Testbed and Participants. To consider a realistic and functional smart home, we deploy the experimental platform in an apartment to collect user usage data of devices to construct our smart home user behavior dataset (AN). We invited three volunteers to simulate the real life of a common family. The volunteers acted as an adult male, an adult female and a child. Our experimental platform contains a total of 36 popular devices on the market and the deployment of these devices is shown in Fig. 12.

Data Collection. The volunteers live in apartments and live and use equipments with to their own habits. During the data collection period, we do not interfere with the volunteers' behaviors. The volunteers are asked to regularly log their behaviors. After the data collection process is over, we check the device usage logs on the smart home apps, and combine the user's behavior records to obtain a user behavior data set. To avoid bias during settle-in period, we let the volunteers live in the experimental environment for least two weeks before

formally collecting data. All users have full knowledge of the IoT devices and the apps used. The volunteers are informed in advance that the usage of the devices will be reviewed and used by us in the future.

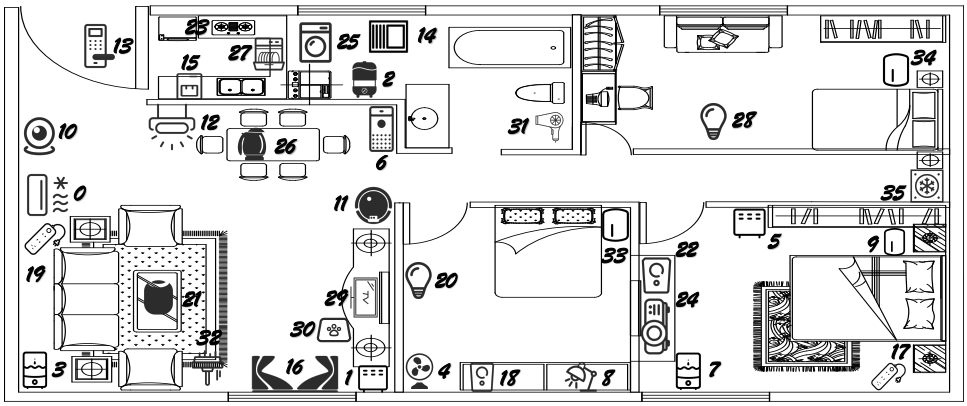


Fig. 12. Overview of testbed setup.

Table 4. Device Information

No.	Device	No.	Device	No.	Device	No.	Device
0	AC	9	bedight_1	18	audio	27	dishwasher
1	heater	10	camera	19	plug	28	bulb_1
2	dehumidifier	11	sweeper	20	bulb_2	29	TV
3	humidifier_1	12	LED	21	soundbox_1	30	pet_feeder
4	fan	13	locker	22	soundbox_2	31	hair_dryer
5	standheater	14	bathheater	23	refrigerator	32	window_cleaner
6	aircleaner	15	water_cooler	24	projector	33	bedlight_2
7	humidifier_2	16	curtains	25	washing_machine	34	bedlight_3
8	desklight	17	outlet	26	kettle	35	cooler

C INTENT INFORMATION OF DEVICES

We describe the intent information of devices used in the intent-aware encoding process in the main body of the paper below. Four tables describe the intent information we annotated for devices in different datasets (AN, FR, SP and US), respectively. Note we consider leaving and returning as one intent of the users.

Table 5. The Intent Information of Devices in Dataset AN

Intent	Devices
Entertainment	TV, Projector, AC, Pet_feeder
Shower	Standheater, Bathheater, Curtains, Hair_dryer
Sleep/getup	Dehumidifier, Bedlight_1, Bedlight_2, Bedlight_3, Bulb_1, Bulb_2
Study	Fan, Desklight, LED
Leave/return	Camera, Locker
Cooking	Refrigerator, Kettle
Cleaning	Sweeper, Washing_machine, Window_cleaner, Dishwasher
Others	Heater, Humidifier_1, Aircleaner, Humidifier_2, Outlet, Audio, Plug, Soundbox_1, Soundbox_2

Table 6. The Intent Information of Devices in Dataset FR

Intent	Devices
Entertainment	Computer, Fan, NetworkAudio, Television, Projector
Cooking	Microwave, Oven, Refrigerator
Washing	Dryer, Dishwasher, Washer
Cleaning	ClothingCareMachine, RobotCleaner,
Leave/return	Camera, Elevator, GarageDoor, MotionSensor, PresenceSensor, SmartLock, Siren
Lighting	Blind, Light
Operational	AirConditioner, AirPurifier, WaterValve, Switch, Thermostat, SmartPlug, RemoteController
Others	ContactSensor, CurbPowerMeter, None, Other

Table 7. The Intent Information of Devices in Dataset SP

Intent	Devices
Entertainment	Computer, Fan, NetworkAudio, Television, SetTop
Cooking	Microwave, Refrigerator
Washing	Dryer, Dishwasher, Washer
Cleaning	ClothingCareMachine, Dryer, RobotCleaner
Leave/return	Camera, Elevator, GarageDoor, MotionSensor, PresenceSensor, SmartLock, Siren, Doorbell
lighting	Blind, Light, Vent
Operational	AirConditioner, AirPurifier, WaterValve, Switch, Thermostat, SmartPlug, RemoteController, Humidifier
Others	ContactSensor, None, Other, GasValve, LeakSensor, MultiFunctionalSensor

Table 8. The Intent Information of Devices in Dataset US

Intent	Devices
Entertainment	Computer, Fan, NetworkAudio, Television, SetTop, Projector
Cooking	Microwave, Refrigerator
Washing	Dryer, Dishwasher, Washer
Cleaning	ClothingCareMachine, RobotCleaner
Leave/return	Camera, Elevator, GarageDoor, MotionSensor, PresenceSensor, SmartLock, Siren, SecurityPanel, SoundSensor
Lighting	Blind, Light, Vent
Operational	AirConditioner, AirPurifier, WaterValve, Switch, Thermostat, SmartPlug, RemoteController, Humidifier
Others	ContactSensor, None, Other, GasValve, LeakSensor, MultiFunctionalSensor, SmokeDetector, Irrigation