IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 1, 2021, accepted April 26, 2021, date of publication May 14, 2021, date of current version May 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3080288

A Stacked Denoising Autoencoder and Long Short-Term Memory Approach With Rule-Based Refinement to Extract Valid Semantic Trajectories

YOGA YUSTIAWAN[1], HANI RAMADHAN[1], AND JOONHO KWON[2]
[1]Department of Big Data, Pusan National University, Busan 46241, South Korea
[2]School of Computer Science and Engineering, Pusan National University, Busan 46241, South Korea

Corresponding author: Joonho Kwon (jhkwon@pusan.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) by the Ministry of Education under Grant NRF-2020R1I1A3072457, in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant by the Korean Government through Ministry of Science and ICT (MSIT) (Development of data improvement and dataset correction technology based on data quality assessment) under Grant 2020-0-00121, and in part by the Ministry of Science and ICT (MSIT) through the Information Technology Research Center (ITRC) Support Program supervised by the Institute of Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2020-0-01797.

ABSTRACT Indoor location-based services have been widely investigated to take advantage of semantic trajectories for providing user oriented services in indoor environments. Although indoor semantic trajectories can provide seamless understanding to users regarding the provided location-based services, studies on the application of deep learning approaches for robust and valid semantic indoor localization are lacking. In this study, we combined a stacked denoising autoencoder and long short term memory technique with a rule-based refinement method applying a rule-based hidden Markov model (HMM) to perform robust and valid semantic trajectory extraction. In particular, our rule-based HMM approach incorporates a direct set of rules into HMM to resolve invalid movements of the extracted semantic trajectories and is extensible to various deep learning techniques. We compared the performance of our proposed approach with that of other cutting-edge deep learning approaches on two different real-world data sets. The experimental results demonstrate the feasibility of our proposed approach to produce more robust and valid semantic trajectories.

INDEX TERMS Deep learning, indoor localization, the Internet of Things, rule-based refinement, semantic trajectories.

I. INTRODUCTION

Location based services (LBSs) provide various services to consumers based on their locations. Recently, the increasing indoor-based services have attracted the attention of researchers to study indoor localization approaches. Because Global Positioning System (GPS) signals cannot be accessed well in indoor environments, various other signals such as radio frequency identification (RFID) [1], magnetic and light sensors [2], [3], Wi-Fi [4], [5] or Bluetooth low energy (BLE) beacons [6]–[10] have been employed for this purpose. Among all approaches, most studies [7]–[12] have focused on the received signal strength indicator (RSSI) obtained from BLE beacons because of the low cost and compatibility for simultaneous scans.

The associate editor coordinating the review of this manuscript and approving it for publication was M. Anwar Hossain.

Fingerprinting-based localization technique is an eminent approach that employs a set of RSSI values from various deployed BLE beacons to conduct indoor localization. Several traditional machine learning approaches that employ fingerprint-based techniques include the k-nearest neighbor (KNN) approach [9], [11]–[13], which employs the location of the K-nearest candidate matching with the fingerprint information to estimate the locations, and the support vector machine (SVM) approach [14]–[16] which aims to identify the location by computing the interval between classes. Hitherto, RSSI observations collected using fingerprints fluctuate violently and have very wide irregular gaps between each value in noisy indoor environments. Moreover, many beacons can be missed in the RSSI observations, thus resulting in the nearby beacon signals remaining undetected for a particular location. Therefore, every location has a particular set of RSSI values in the fingerprints. The critical issue is providing

robust performance when the received signals are unsteady over time.

These traditional machine learning approaches cannot learn any relationship and structure between the RSSI values and the location to provide robust indoor localization. Recently, deep learning approaches have emerged as the most highly sought techniques for numerous applications. However, deep learning has not been extensively applied in the field of indoor localization yet. Denoising autoencoder (DAE)-based approaches [17] can learn stable and robust representation from the noisy RSSI-based datasets. In [18], a deep neural network (DNN) was applied for floor-level estimation based on Wi-Fi fingerprints; however, it could not learn the temporal dependency of movements. In another study [19], a recurrent neural network (RNN) was applied to establish a deep mapping relationship between fingerprints and locations over time; however, this too suffered from the gradient loss problem, which significantly affected performance. The long short term memory (LSTM) architecture is widely known as a deep learning approach that can prevent the gradient loss problem and can capture the temporal dependency of movements. The previous LSTM-based methods [2], [20] have indicated robust performance; however, these methods have not been tested on RSSI-based datasets. One approach [21] highlighted the robustness in estimating coordinate positions with some acceptable errors. Therefore, a combination of DAE and LSTM can be utilized to provide robust indoor localization which can overcome the issues associated with noisy datasets and can learn the temporal dependency of datasets.

Most current deep learning based approaches [3], [17], [22], [23] extract indoor trajectories based on a set of coordinate positions. However, the coordinate positions may not be helpful in providing indoor location-based services. For instance, in an exhibition, a recommender system cannot recommend the exhibit based on coordinate positions (e.g. exhibit A at (12,5)), which would be difficult for visitors to understand. Therefore, it will be better if the usual indoor location is enriched by semantic information to provide seamless understanding for visitors (exhibit A is located in the Classical Area). A sequence of semantic locations is then regarded as a semantic trajectory.

Semantic trajectories accumulate the essence of human movement and help understand the movement patterns easily. Semantic-enriched movement such as (Classical Area → Contemporary Area) presents a very useful solution to real-world problems in many cutting-edge location-based applications [24]–[26]. In an indoor environment, the semantic positions can be represented simply as an indoor area of real-world cases, e.g., an abstract area in an art exhibition.

The extracted semantic trajectories from indoor localization techniques can become invalid because of a noisy environment and unsteady received signals. For example, a user cannot move from one location to another distant location without passing the location in between or within a short time. Figure 1 illustrates two semantic trajectories which show

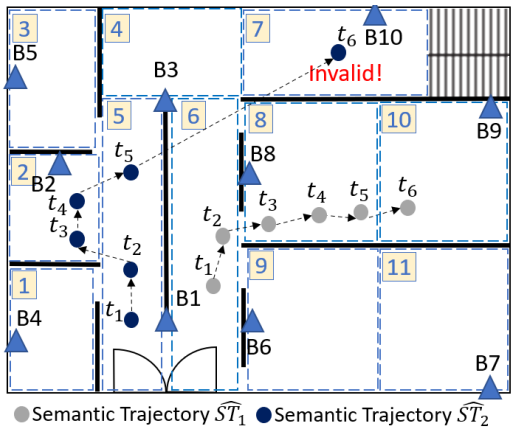

FIGURE 1. Valid and invalid movements.

invalid and valid movements. The first semantic trajectory $\hat{ST}_1$ shows a valid movement, whereas the second semantic trajectory $\hat{ST}_2$ displays an invalid movement. The invalid movement occurs between timestamp t_5 and t_6. This consecutive movement jumps from semantic location 5 to 7 without passing location 4, which is impossible in a real-world case. Not a lot of studies have focused on this issue. The constraint-based approach [7] discussed the validity of the extracted semantic trajectory in indoor environments. However, it heavily relies on the inferred semantic graph from the floor plan rather and less considers the noisy characteristic of the RSSI signal. Another study [4] introduced a fingerprint graph to a hidden Markov model (HMM) that can reduce the candidate states and set the edges as a walkable area. Instead, a rule-based HMM can also be utilized to estimate the movement by setting the defined set of rules into the HMM for resolving this invalid extracted semantic trajectory. This can validate an invalid movement within the initial extracted semantic trajectory, thus, the extracted semantic trajectory becomes valid while maintaining its robustness.

In this study, we propose training a deep learning frame which combines stacked DAE and LSTM (DSLSTM) and then introduces rule-based refinement that applies a rule-based HMM approach to produce robust and valid semantic trajectory extraction. The DSLSTM-based approach can handle the sequence of RSSI observations in noisy indoor environments and can learn the temporal dependency of the trajectories. Furthermore, the rule-based HMM approach uses a set of rules for the state transition inside the HMM, which generates valid movements in extracted semantic trajectories. To the best of the author's knowledge, our work is the first to utilizes a deep learning based approach and rule-based HMM for RSSI-based indoor localization. We performed experiments based on our real-world dataset directly collected using BLE beacons, and the publicly available real-world dataset obtained from [27]. Our experimental results indicate that the proposed approach extracts valid semantic trajectories and outperforms the other cutting-edge approaches. Additionally, the proposed rule-based approach

is extensible to other deep learning-based approaches for valid semantic trajectory extraction.

The key contributions of this paper can be summarized as follows:

- We adopt a novel deep learning-based approach that combines stacked DAE and LSTM to perform robust semantic trajectory extraction.
- We introduce a rule-based refinement technique that applied the rule-based HMM approach to extract valid semantic trajectories using a set of rules applied in the HMM to represent valid movements.
- The proposed rule-based HMM approach is the first to be studied and is extensible to be applied with other deep learning-based approaches for valid semantic trajectory extraction.
- We performed a comprehensive experimental evaluation of our proposed approach with current state-of-the-art deep learning based indoor localization techniques using the collected, and publicly available real-world BLE RSSI measurement dataset. The experimental results demonstrate the advantages of the proposed approach.

The remainder of this paper is organized as follows. Chapter II details the related research work. Chapter III explains the basic knowledge of this work. Chapter IV describes the proposed approach. Chapter V presents the details of the experimental design, the results, and a discussion of the experimental results. Finally, we conclude this paper in Chapter VI.

II. RELATED WORK

Herein, we briefly review some related works, and compare them with our proposed method.

A. INDOOR LOCALIZATION METHODS

Indoor localization has been extensively investigated in recent years. Several approaches have been developed using various machine learning methods such as KNN [28]–[30], SVM [16], [31], and tree-based methods [5], [32], [33]. The improved weighted KNN [29] introduces the Manhattan distance to the KNN approach to enlarge or reduce the differences in RSSI in each dimension to obtain the influence of each reference point. Another KNN-based approach [30] aims to eliminate the incorrect neighboring reference points, which increases the efficiency and accuracy of clustering. The i-KNN approach [28] attempts to filter the initial datasets by considering the proximity between RSS fingerprints to the BLE beacons. However, these KNN-based methods achieve a low positioning error of approximately 6 m. Furthermore, the fuzzy least squares, LS-SVM-based approach [31] aims to mitigate the impact of noise and outliers, while another SVM-based approach [16] aims to construct a hyperplane with the maximal margin between different classes which decrease the number of misclassifications. These SVM-based methods also suffer significant loss of discriminative information, which results in positioning errors of approximately 5 meters. In contrast, two random forest-based studies [32], [33] achieved satisfying performance in terms of accuracy; however, these approaches have not been tested yet on highly noisy indoor environments. A deep fuzzy forest [5] addressed noisy data and traces the human and robot trajectory with Wi-Fi fingerprinting but only works in the cartesian coordinate instead of semantic space. Some HMM-based methods [8], [34] have exhibited good performance but only for a subset of observations. Therefore, because our aim is to support the processing of semantic trajectories in a real-time manner, these two methods are not applicable.

B. DEEP LEARNING-BASED INDOOR LOCALIZATION METHODS

Deep learning-based approaches have been studied to provide robust indoor localization in noisy environments. Several state-of-the-art DNN-based approaches [3], [18], [35]–[37] have been shown to produce accurate indoor localization with certain minimum loss; however, DNN cannot learn the temporal dependency of sequential fingerprint sets. A tracking system utilized [3] LiDar and color features to extract the 3D trajectory of human movement to be tracked by a moving robot. The tracking system uses convolutional neural network (CNN), a deep learning technique, to identify the moving people thus follows them. However, the CNN is only limited to recognize the human position in the image while the majority part of real time indoor positioning is determined by other image and signal processing techniques. Auto-encoder based methods [17], [23], [38] aim to learn the robust and stable representation of RSSI fingerprints in noisy indoor environments; however, they cannot estimate indoor locations. A seq2seq deep learning model [10] showed a framework to predict and analyze visitor paths in a museum from non-invasive bluetooth monitoring system where each sensor placed far apart. Additionally, an LSTM-based approach [21] successfully estimates the indoor locations from RSSI fingerprints and achieved a robust performance. However, [10], [21] did not address the issue of invalid semantic trajectories, wherein movement occurs from one location to a distant location within a short time. This movement is impossible in real-world cases; therefore, our work focuses on producing valid semantic trajectories.

C. INDOOR SEMANTIC TRAJECTORY EXTRACTION

Recent studies [7], [9], [39]–[41] investigated the extraction of semantic positions in the indoor environment and its extension. The multi-slot BLE positioning system [9] estimates both cartesian and semantic-labeled positions in a mixed indoor and outdoor environment. Although the output is not necessarily represented as trajectory, the system takes consecutive inputs of raw BLE RSSI and returns the position within each timestamp. Moreover, the positioning uses nearest neighbor algorithm which is highly prone to error. A constraint-based approach [7] discussed the extraction of the valid semantic trajectories in the indoor environment using BLE RSSI. The approach employed the semantic graph of the floor plan as the constraints to ensure that the

estimated position is close to the neighbors of the previously inferred position. However, this approach merely utilized the aggregation technique such as max and mean function to remove the noise of the unstable RSSI and only extended to a simple neural networks as the most complex structure. Thus, we may acquire less robust valid trajectories using the simple aggregation technique and machine learning models. A region-based rule approach with cleaning and recovery [39] extracted semantic trajectory from raw Wi-Fi positioning data. This approach, nonetheless, still requires the 2D/3D position to associate with the semantic position. On the other hand, the network-embedding and FP-growth [40] extracts the association rule from the Wi-Fi indoor positioning-based semantic trajectories. Nonetheless, the discovered association rule may not be fully maximized because the size of trajectory dataset is highly reduced due to positioning errors and bad data quality. An indoor-outdoor spatial representation unification [41] worked on the matching human movement in indoor and outdoor space to the semantic trajectory. The representation successfully models the semantic trajectory but only extend the work to the query and not detailed in semantic trajectory extraction.

D. THE PROPOSED METHOD

We aim to solve invalid movements when extracting semantic trajectories. We introduced the combined stacked denoising autoencoder with a rule-based refinement method that applies direct rule-based HMM to ensure movement occurs from one location only to its possible adjacent locations based on floor-map representations. The set of rules are applied with the HMM to encourage the HMM to learn valid movements. Therefore, HMM can output valid state transitions where each state is regarded as a semantic location. The refinement process by HMM supports continuous processing of semantic trajectories within a particular tumbling window interval.

III. PRELIMINARIES

In this section, we present the notions of the semantic indoor trajectory and the characteristics of signals of BLE beacons.

A. SEMANTIC TRAJECTORIES

Semantic locations provide additional semantic information regarding the respective location over coordinate-based locations. In the simplest form, an indoor semantic location is described by semantic labels that represent particular indoor areas that can be understood by users, e.g., ''classical area'', ''contemporary area'', and ''abstract area''. In [42], semantic trajectories are defined as sequences of timestamped locations that are described by the indoor area and semantic labels. Semantic trajectories capture human movement patterns at a high conceptual level for better understanding. Equation 1 describes a semantic trajectory consisting of sequences of timestamped semantic locations with $t_i < t_{i+1}$.

Example 1: Figure 2 depicts an example of the floor plan that includes 11 semantic locations and 10 deployed beacons. Location 5 *is a history corridor, whereas location* 6 *represents a charity corridor.*

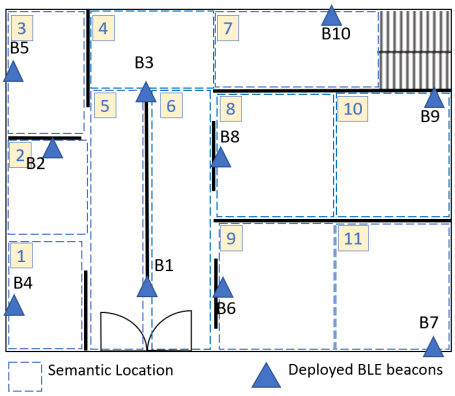

FIGURE 2. An example of floor plan.

$$ST = \{(t_1, s_1), (t_2, s_2), \ldots, (t_i, s_i)\} \tag{1}$$

B. FINGERPRINTS

The fingerprinting approach mainly aims to use RSSI measurements from all deployed BLE beacons across the indoor environment to build a fingerprint dataset. A typical fingerprint data set can be illustrated as follows:

$$F = \{(f_1, s_1, t_1), (f_2, s_2, t_2), \ldots, (f_i, s_i, t_i)\} \tag{2}$$

where f_i denotes the fingerprint pattern and s_i is the semantic location recorded at time t_i. The fingerprint pattern f includes the raw RSSI measurements from all deployed beacons. Therefore, f can be described as follows:

$$f_i = (rss_1, rss_2, \ldots, rss_m) \tag{3}$$

where m is the total number of deployed BLE beacons and rss_i depicts the measured RSSI value from the i^{th} beacon. The RSSI value is the negative value within the range of $(-1, -100)$. If the RSSI value of the i^{th} beacon is almost zero, it means that the user's location is near the i^{th} beacon, and vice versa. An undetected beacon signal is defined by a zero value. The fingerprinting technique often suffers from insufficient RSS measurements caused by noisy indoor environments. The RSSI value may violently fluctuate, and gaps for each observed value may vary widely; therefore, it is hard to conduct accurate localization using raw RSSI measurements. Another problem occurs when the observable beacons from the user's location fail to send RSSI values with the observations.

Example 2: Based on Figure 2, suppose a user is located at location 10; therefore, two beacons (B8 & B9) are actually observable to send signals. However, one beacon in B9 fails to send an RSSI value in the observation. This missed beacon can deteriorate the performance of indoor localization.

C. LONG SHORT TERM MEMORY (LSTM)

The vanilla LSTM architecture is known as a basic LSTM architecture. It is considered as a simple recurrent architecture because the output gate forms a loop within the network [21].

The LSTM cell obtains the cell state and hidden output of the current timestamp and uses them for the next timestamp. Therefore, the previous state is utilized for the next estimation. The main components of the LSTM are the cell state and its various gates. The cell state acts as a transport medium that passes relative information all the way down the sequence chain, whereas the gates are different neural networks (NNs) that decide which information is allowed on the cell state. The gates can learn the information that is relevant to keep or forget during training. An LSTM has three different gates—forget gate, input gate, and output gate. The vanilla LSTM architecture is depicted in Figure 3.

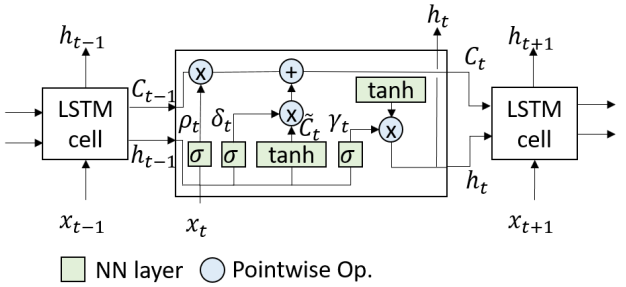

FIGURE 3. Vanilla LSTM architecture.

The first step is to decide which information should be thrown away or kept, which is performed by the forget gate ρ_t. The forget gate utilizes a sigmoid layer. It investigates the input h_t and x_t and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A value of 1 means that the information should be kept, whereas 0 means that the information should be thrown away. The forget gate is denoted in Equation 4, where b is a bias and W denotes weights.

$$\rho_t = \sigma(W_{rho} \cdot [h_{t-1}, x_t] + b_\rho) \tag{4}$$

The second step is to decide which new information must be stored in the cell state. We utilize the input gate δ_t that decides which values will be updated by transforming the values to be between 0 and 1. A value of 0 means the information is not important, and a value of 1 means it is important. The input gate uses a sigmoid layer. A tanh layer makes a vector of new candidate values $\tilde{C}_t$ that could be added to the state. Next, we combine these two outputs from a sigmoid and tanh layer to create an update to the state. The input gate is represented in Equation 5.

$$\delta_t = \sigma(W_\delta \cdot [h_{t-1}, x_t] + b_\delta) \tag{5}$$

While the candidate values are computed as in Equation 6.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{6}$$

Now, we have to update the old cell state C_{t-1} into the new cell state C_t. We multiply the old state by ρ_t to forget things that we decided to forget earlier. Then, we multiply the candidate values by δ_t to decide how much information must be updated for each state value. The new cell state C_t is given as shown in Equation 7.

$$C_t = \rho_t \times C_{t-1} + \delta_t \times \tilde{C}_t \tag{7}$$

The last step is to decide which hidden state should be next. This is performed by the output gate γ_t. The output gate exercises the sigmoid layer. Then, we put the cell state C_t through tanh and multiply it by the output of the forget gate; thus, we only obtain an output with the required parts. The output gate is expressed as shown in Equation 8.

$$\gamma_t = \sigma(W_\gamma \cdot [h_{t-1}, x_t] + b_\gamma) \tag{8}$$

The next hidden state is represented using Equation 9.

$$h_t = \gamma_t \times \tanh(C_t) \tag{9}$$

IV. PROPOSED SYSTEM

In this section, we present a detailed description of our approach for extracting semantic trajectories from the deployed BLE beacons.

A. SYSTEM ARCHITECTURE OVERVIEW

Figure 4 depicts the system architecture of our proposed DSLSTM and rule-based refinement module. Our system comprises two phases—an offline phase and an online phase. The inputs for our system are the fingerprint sets that are the labeled and unlabeled sets. The labeled training set is mainly used for the learning of the robust DSLSTM model in the offline phase; however, it is also utilized for learning HMM refinement. Whereas the unlabeled testing sets are primarily used for continuous valid semantic trajectory extraction in the online phase and learning the HMM refinement. Both training and testing is preprocessed using data normalization to convert the range of input into the scale (0-1). The offline phase aims to learn and obtain a robust DSLSTM model and outputs the robust DSLSTM model as an input for location estimation in the online phase. The online phase aims to validate the estimated semantic location using the HMM refinement module which applies the direct rule-based

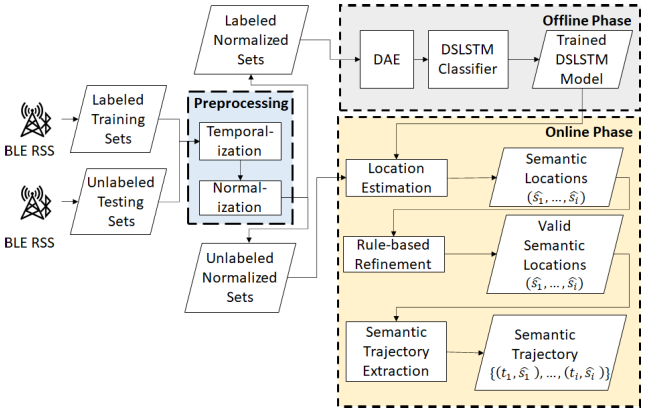

FIGURE 4. System architecture of our proposed approach.

HMM approach; thus, it can produce valid movement for semantic trajectory extraction.

B. PREPROCESSING

Before we feed the labeled fingerprint into the LSTM models, we conduct a series of preprocessing operations including temporalization and data normalization.

LSTM models are known as deep learning recurrent models, and significant attention is required for data preparation. The input data for an LSTM model comprises a 3D array, which has the shape of *samples* × *lookback* × *features*. The LSTM model aims to learn the temporal dependency of fingerprints over time. Therefore, samples are the number of observations or entries, and features are the number of deployed beacons across the indoor environment. In contrast, lookback means that at time t_i, the LSTM requires to look at the past data up to $(t_i - lookback)$ to estimate the location. The input data from raw fingerprints is a 2D array of size *samples* × *features*. Consequently, we need to transform the 2D input to the required 3D input *samples* × *lookback* × *features*. This transformation process is conducted through a temporalization module inside the preprocessing module. The raw fingerprint set $F = \{(f_1, t_1), (f_2, t_2), \ldots, (f_i, t_i)\}$ is transformed to the new fingerprint set such as $F = \{\langle (f_{i-lb+1}, t_{i-lb+1}), \ldots, (f_i, t_i)\rangle, \ldots\}$. The temporalization process is summarized in Algorithm 1.

Example 3: Figure 5 illustrates an example of temporalization using five deployed BLE beacons with tumbling window size 3. We set the lookback to 2 because the lookback should be long enough to contain the history of the trajectory but not too long to burden the computation time. In this example, using 2 as lookback value is enough to capture the previous information in a small data. Note that, for bigger rooms and longer trajectories, the lookback value can be set to larger than 2. Thus, we process the current and one previous fingerprint data t_{i-1} at the current time t_i. Here, we have four fingerprints in the fingerprint set. At each fingerprint, we have RSSI values from five beacons $f_i = (rss_1, rss_2, \ldots, rss_m)$. The first three tuples are shown as $\langle(-98, -85, 0, -76, -55), (0, -76, -55, -72, -88), (-50, -71, 0, 0, -48)\rangle$. *There is no overlapping window in the tumbling window; thus, we slide the current tumbling window from the first three tuples to the next tuples without overlapping with the first three tuples. For each temporalization tumbling window, we transform the 2D array into a 3D one for every tuple. Therefore, within the window, the three tuples* $\langle(-98, -85, 0, -76, -55), (0, -76, -55, -72, -88), (-50, -71, 0, 0, -48)\rangle$ *are processed to become the temporalized ones* $\langle(-98, -85, 0, -76, -55), (0, -76, -55, -72, -88)\rangle, \langle(0, -76, -55, -72, -88), (-50, -71, 0, 0, -48)\rangle$. *Furthermore, we apply the same procedure to the next temporalization tumbling window.*

As described in Section III-B, the fingerprint set often suffers from noisy and undetected RSSI measurements. Therefore, many outliers exist that possibly occur because of the initial fingerprint set. The RSSI measurements are usually

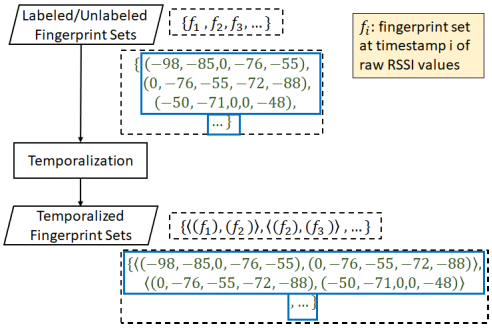

FIGURE 5. Execute temporalization from unlabeled/labeled fingerprint sets.

Algorithm 1: Transformation of the Initial 2D Shape of the Fingerprint Inputs Into a 3D Shape

Input : The 2D shape fingerprint inputs *input_X* and target location *input_Y*
Output: The 3D shape of *X* & *Y*

```
Function Temporalize(X,Y,lookback)
    Initialize empty list of X
    Initialize empty list of Y
    for i in range(totalsamples − (lookback − 1)) do
        Initialize empty list of temporals t
        for j in range(0, lookback) do
            // Gather the past samples up
               to lookback period
            Append the input training input_X up to
             lookback period into list t
        end for
        Append t to the list input_X
        Append the target location input_Y at the time
         i + lookback − 1
    end for
    return X, Y
```

in the range of [-100,0]; the scale is wide and may affect the performance of training of the deep learning model. To resolve this issue, we employ a data normalization tumbling window by count to reduce the complexity of the RSSI measurements in the fingerprint set that the neural network is trying to learn. This can potentially reduce the loss of the model and accelerate training. Data normalization can also enhance the network to converge and save the network from scaling issues. Data normalization can only be applied to the fingerprint pattern f_i that contains RSSI measurements from all deployed beacons. Therefore, using a lookback equal to 2, we can modify the fingerprint set from $F = \{\langle (f_{i-lb+1}, t_{i-lb+1}), \ldots, (f_i, t_i)\rangle, \ldots\}$ into the new form of fingerprint set $F = \{\langle (f_1), (f_2)\rangle, \ldots, \langle (f_{i-lb+1}), (f_i)\rangle\}$. This means we consider the current f_i and previous fingerprint f_{i-1} at time t_i as one tuple $\tau_1 = \langle (f_1), (f_2)\rangle$ for lookback $lb = 2$. The normalization tumbling window with window size l executes the normalization function to the raw fingerprint tuple τ_i. The number of tuples τ_i is normalized

within a specific window size l. Because the tumbling window incorporates a non-overlapping window, the tuple τ_i that belongs to the previous tumbling window will not be in the current tumbling window. Then, the normalization function on the tuple τ_i yields a normalized fingerprint set $\overline{F} = \{\langle(\overline{f}_1), (\overline{f}_2)\rangle, \ldots, \langle(\overline{f}_{i-lb}), (\overline{f}_i)\rangle\}$. The data normalization can be denoted using the following equation:

$$\overline{f}_i = \frac{(f_i - \min(f_i))}{(\max(f_i) - \min(f_i))} \tag{10}$$

Example 4: Figure 6 illustrates an example of normalization using five deployed BLE beacons with the same tumbling window size and lookback of 3 and 2, respectively. At each fingerprint, we have RSSI values from five beacons $f_i = (rss_1, rss_2, \ldots, rss_m)$. *The first tuple is shown as* $\langle(-98, -85, 0, -76, -55), (0, -76, -55, -72, -88)\rangle$, *and the second is* $\langle(0, -76, -55, -72, -88), (-50, -71, 0, 0, -48)\rangle$. *There is no overlapping window in the tumbling window; thus, we slide the current tumbling window from the first two tuples* $\langle(f_1), (f_2)\rangle$ *and* $\langle(f_2), (f_3)\rangle$ *to the next tuples without overlapping with the first two tuples. For each normalization tumbling window, we apply the Equation 10 on every tuple. Therefore, within the window, the two tuples* $\langle(-98, -85, 0, -76, -55), (0, -76, -55, -72, -88)\rangle$, $\langle(0, -76, -55, -72, -88), (-50, -71, 0, 0, -48)\rangle$ *are processed to become the normalized ones* $\langle(0, 0.13, 1, 0.22, 0.44), (1, 0.14, 0.38, 0.18, 0)\rangle$, $\langle(1, 0.14, 0.38, 0.18, 0), (0.30, 0, 1, 1, 0.32)\rangle$. *Furthermore, we apply the same procedure to the next normalization tumbling window.*

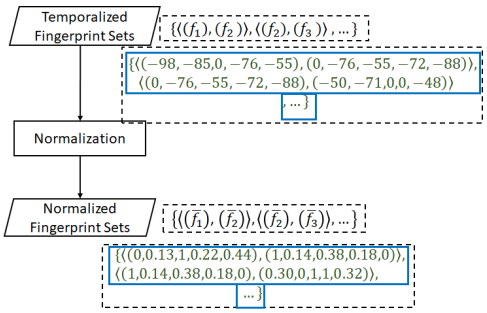

FIGURE 6. Execute normalization from temporalized fingerprint sets.

C. OFFLINE PHASE

In the offline phase, we utilize the labeled training sets which is the fingerprint with known semantic location for each timestamp to obtain a robust DSLSTM model. Initially, we have a fingerprint given as $F = \{(f_1, s_1, t_1), (f_2, s_2, t_2), \ldots, (f_i, s_i, t_i)\}$; then, we preprocess the fingerprint set using the same procedure explained in the Section IV-B with slightly different steps to obtain the normalized fingerprint set. We split the fingerprint set F to the training input X and target location Y. The training input is modified to $X = \{\langle(f_{i-lb+1}), \ldots, (f_i)\rangle, \ldots\}$ and the target location becomes $Y = \{\langle s_1\rangle, \ldots, \langle s_i\rangle\}$, where $x_i = \langle(f_{i-lb+1}), (f_i)\rangle$ and $y_i = \langle s_i\rangle$. Then, we apply data normalization to the training input, resulting in $X = \{\langle(\overline{f}_{i-lb+1}), \ldots, (\overline{f}_i)\rangle, \ldots\}$. After we have the normalized training input, we are ready to input the normalized sets to the deep learning (DSLSTM) network. The deep learning network learning process comprises two main modules, the DAE and the DSLSTM classifier.

1) DENOISING AUTOENCODER (DAE)

The proposed deep learning framework is built from a combination of two well-known deep-learning based methods—stacked DAE and LSTM. First, we introduce the stacked DAE, which is an unsupervised NN to recover the input. Now, we get the normalized fingerprint set $\overline{F}$, which can be sent to the DAE model for denoising. For better understanding, now we change the notation for normalized fingerprint set $\overline{F}$ to X. Denoising here means that the fingerprint will be trained to learn the robust representation E from noisy and corrupted fingerprint set X. The stacked DAE process consists of noise injection, encoding, and decoding. Our proposed DAE model takes the input X and corrupts the input using the Dropout method to increase robustness. The encoding process uses forward propagation to calculate the activation function of multi-layers and brings the input fingerprint set to a lower dimension which is the encoded one E. Finally, the decoding process takes the encoded input E and maps it back to the robust reconstructed input $\widehat{X}$. Then, we use the encoded input E to be the input for the stacked LSTM (SLSTM) classifier model. The architecture of the DAE model is illustrated in Figure 7.

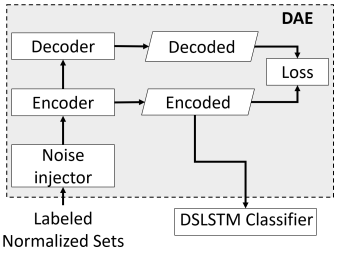

FIGURE 7. Architecture of the stacked denoising autoencoder (DAE).

The DAE model is trained to achieve a robust DAE model to estimate the location accurately. The noisy input X will be corrupted when the dropout $\mathrm{dp}(X)$ is used to achieve robustness when training the model and to prevent overfitting that may occur during training. The corrupted input will be fed to the encoder $\mathrm{en}(\mathrm{dp}(X))$ to obtain the encoded input E. The decoder $\mathrm{de}(E)$ maps the encoded input E back to the reconstructed input $\hat{X}$. This DAE process is represented by Equation 11.

$$\widehat{X} = \mathrm{de}(\mathrm{en}(\mathrm{dp}(X))) \tag{11}$$

Given input X, the DAE trains the adjustments of noise injection, encoding, and decoding to minimize the reconstruction error, which is aligned to minimize the following

objective function in Equation 12.

$$J_{DAE}(X, \hat{X}) = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \hat{x}_i\right)^2 \tag{12}$$

The encoded input E and the decoded input D are fed into the loss module to compute the reconstruction error, as in Equation 12. The reconstruction error operates on the normalized RSS rather than the raw RSSI to ensure that the model does not overfit on the noisy raw RSSI values. Then, the DAE model with minimal reconstruction error is chosen, and the encoded input is utilized as the input for the DSLSTM classifier.

2) DSLSTM CLASSIFIER

We build and combine the stacked LSTM (SLSTM) to the established DAE model (DSLSTM classifier) for training the robust representation input E; thus, finally, we can obtain a robust model to estimate the location and extract the trajectory accurately. We use LSTM because it can learn the temporal dependency between locations at each time, which is crucial in the case of semantic trajectory data. Therefore, LSTM takes the 3-dimensional input which considers not only the number of samples and features but also the timestep or lookback for each sample. The lookback lb aims to include the $lb-1$ past data for each sample at time t_i and the sample itself as the input; therefore, it considers the dependency between each sample at different times. We establish the stacked LSTM model in which several LSTM layers are stacked to be trained. The main part is the architecture of our DSLSTM model, which is depicted in Figure 8.

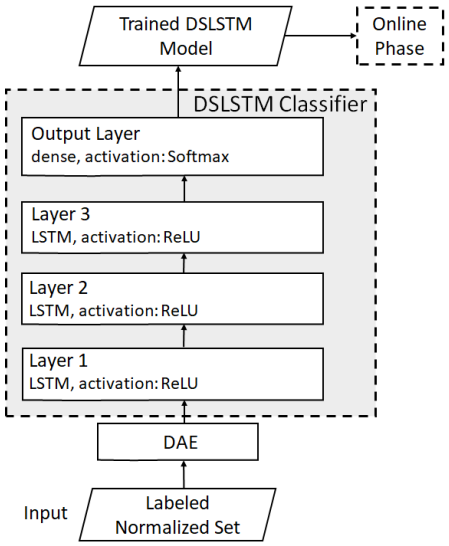

FIGURE 8. Architecture of DAE-LSTM.

Until this part, we have a robust DAE model with minimal reconstruction error. We then combine the DAE model with the SLSTM model to construct the DSLSTM classifier. We consider the robust representation or encoded input E as the output from the DAE model and the input to the DSLSTM classifier model. The actual semantic location Y of the past semantic trajectory is exercised to train the supervised classification layer which estimates semantic locations $\hat{Y}$ and extracts the semantic trajectory. The target location Y must be converted using one hot encoding. One hot encoding is a representation of categorical values as binary vectors. For example, we have seven locations that can be regarded as seven classes; thus, $y_i = 2$ will be converted to binary vector $y_{i,c} = [0, 1, 0, 0, 0, 0, 0, 0]$. Each integer value is represented as a binary vector that has all zero values except the index of the integer, which is marked as 1. The last output layer using the softmax function activation will produce the probability for each class $p_{i,c}$. To achieve robustness, DSLSTM training aims to minimize the entropy-like objective function in Equation 13. The chosen objective function ensures that each location has the balanced visiting chance purely from the labeled normalized RSSI set in this step. The probability of the transition from each location to other location is discussed later in refinement rule. Furthermore, DSLSTM training also aims to maximize the accuracy acc of estimated classes based on Equation 14.

$$J_{DAE-LSTM} = \frac{1}{n}\sum_{i=1}^{n} y_{i,c}\log\left(p_{i,c}\right) \tag{13}$$

Algorithm 2 describes the complete procedures of the DSLSTM training method. The algorithm considers a set of labeled normalized fingerprints that have the semantic locations for each sample. We consider this set as the normalized training input. The normalized training input is fed to the DSLSTM model; then, we use the actual target location to train the DSLSTM model. During training, we place checkpoints where the training process has minimal loss. Once training is completed, we can obtain a robust DSLSTM model which has the most minimum loss and best accuracy.

$$acc = \frac{total\ number\ of\ correct\ class\ estimation}{total\ number\ of\ class\ estimation} \tag{14}$$

Algorithm 2: DSLSTM Model Training

Input : The labeled normalized fingerprint set $\overline{F}$
Output: The DAE-LSTM model
// Initialization
Set the parameter settings for the DSLSTM network Set *n_epoch* Set *batch_size* // Model training
for *n_epoch and batch_size* **do**
  Train the DSLSTM network
  Calculate *loss_function* using Equation 13
  Set checkpoint *ckpt* where minimal loss occurs
end for
Save the optimal DSLSTM model

D. ONLINE PHASE

In the online phase, we perform semantic trajectory extraction from a continuous unlabeled testing set which contains

fingerprints with unknown semantic locations. In the real-world, the collected fingerprints are based on continuous RSSI observations without knowledge of the indoor environment. Therefore, it is important to estimate robust semantic locations and extract a semantic trajectory. To achieve this purpose, we utilize the robust DAE-LSTM based approach that can estimate the semantic locations accurately. However, when the semantic trajectory is extracted based on the estimated semantic locations, the extracted trajectory may be invalid because of the observations collected in noisy indoor environments. Therefore, we introduce a rule-based refinement process which involves applying rule-based HMM that can avoid invalid movements of the extracted semantic trajectories.

1) ROBUST LOCATION ESTIMATION

By now, we already have a robust DSLSTM model. Thus, we want to utilize the robust DSLSTM model to estimate the semantic locations accurately. First, we receive the unlabeled testing set continuously, which comprises continuous RSSI observations without the semantic locations. Then, we conduct preprocessing using the same procedure described in Section IV-B to the testing input X and return the normalized testing input $\overline{X}$. By utilizing the DSLSTM model, initially, we can directly estimate the semantic locations and extract the semantic trajectory. However, it still may output invalid movements from the previous estimated location to the current estimated location. Invalid movement denotes a movement between one semantic location to another far semantic location within a short time. Invalid movement is impossible to occur in the real world because usually the time interval between each consecutive sample is only within a second. To solve this issue, we can obtain the location estimation as a sequence of semantic locations $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_i$. The sequence of semantic locations can be expressed such as Equation 15. Then, the sequence $\hat{S}$ can be the input for the rule-based refinement module to resolve this issue.

$$\hat{S} = \{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_i\} \tag{15}$$

2) RULE-BASED REFINEMENT

Rule-based refinement aims to extract a valid semantic trajectory that can solve invalid movements of the initial extracted semantic trajectory. As mentioned in the beginning of Section IV-D, we employ HMM to recognize states corresponding to the estimated semantic locations $\hat{S}$. By incorporating the HMM into the rule-based refinement, we can determine invalid state transitions according to the defined set of rules. We call this the rule-based HMM approach.

The sequence of semantic locations is an input to the HMM. We use the HMM with the Gaussian emission (GaussianHMM) model and Viterbi algorithm to predict the most probable state sequence. We use a single Gaussian distribution to represent the state-dependent observation space for an HMM state $\hat{s}_t$. In this case, we estimate the most probable state sequence at time $(1, \ldots, T)$ and most probable state at time $T + 1$, given a state sequence from $(1, \ldots, T)$. Therefore, we fit the GaussianHMM model directly using a set of estimated semantic locations as the output of the DSLSTM model $\hat{S} = \{\hat{s}_1, \ldots, \hat{s}_i\}$ from $i = 1$ until $i = T$. We summarize the description of the components of GaussianHMM of our case in Table1.

TABLE 1. GaussianHMM inside the rule-based refinement.

Symbol	Description
S	Set of semantic position S
A	Transition probability between each semantic position $A(i,j) = P(\hat{s}_i \rightarrow \hat{s}_j)$ where $\hat{s}_i, \hat{s}_j \in S$
B	Emission probability using conditional Gaussian distribution for state $\hat{s}_i \in \hat{S}$: $B(i) = (\mu_i, \Sigma_i) = P(y_t \mid \hat{s}_i)$
π	Initial probability of a semantic position at $t = 1$

First, we set a rule wherein each semantic location has its valid neighboring set *VN* and movement to the semantic location inside *VN* is considered valid. We consider this movement as valid state transition inside GaussianHMM; therefore, we can set the transition probability between each state corresponding to the rule of the movement between each semantic location. The movement from one semantic location to outside of its valid neighboring set is considered invalid. Consequently, if the transition between state $\hat{s}_i$ to state $\hat{s}_j$ is considered invalid, then the transition probability from $\hat{s}_i$ to $\hat{s}_j$ is set to zero. This rule can be defined as in Equation 16.

$$A(i,j) = 0, \text{when } \hat{s}_i \rightarrow \hat{s}_j, \hat{s}_j \notin VN(\hat{s}_i) \Lambda \hat{s}_i, \hat{s}_j \in \hat{S} \tag{16}$$

Furthermore, the emission probability is randomly defined by a conditional Gaussian distribution through fitting the model directly to a set of estimated locations within time interval $(i = 1, \ldots, T)$ as shown in Equation 17

$$P(y_i \mid \hat{s}_i) = N(y_i \mid \mu_{\hat{s}_i}, \Sigma_{\hat{s}_i}) \tag{17}$$

where μ_i, Σ_i are the mean vector and covariance matrix associated with state $\hat{s}_i$ and $y_i \in$ observation sequence Y. Then, $N(y_i \mid \mu_{\hat{s}_i}, \Sigma_{\hat{s}_i})$ is described as a conditional Gaussian distribution in Equation 18, where $\Sigma_{\hat{s}_i} = \text{cov}[Y]$. Furthermore, we build another rule for the rule-based HMM output to satisfy an empirical valid assumption in one case.

$$N(y_i \mid \mu_{\hat{s}_i}, \Sigma_{\hat{s}_i}) = \frac{1}{(2\pi)^{D/2} \mid \Sigma_{\hat{s}_i} \mid^{\frac{1}{2}}} \exp\left(-\frac{1}{2}[\mathbf{y_i} - \mu_{\hat{s}_i}]\Sigma_{\hat{s}_i}^{-1}[\mathbf{y_i} - \mu_{\hat{s}_i}]^\top\right) \tag{18}$$

Example 5: Figure 9 illustrates the state transition of simple Gaussian HMM. Suppose we have a sequence of valid state transition from t_1 to t_3, which is $5 \rightarrow 5 \rightarrow 4$. Then, we would like to decide which one is the most likely state in t_4 given the known valid states between t_1 to t_3. We compute the observed probability of each possible next state at t_4 using Equation 17. Suppose the observed probabilities to the possible state 4, 5, and 6 are 0.49, 0.40, and 0.55, respectively, and the other states such as state 1 are 0 because of the rule where the state transition is invalid. Therefore, the most likely

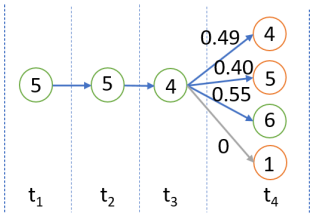

FIGURE 9. Avoiding invalid movement with simple Gaussian HMM because transition from state 4 to state 1 is impossible.

state in t_4 is the one which has the highest probability, which is state 6.

Algorithm 3 demonstrates the procedure for validating the initial extracted semantic trajectory. First, we obtain the input as a set of estimated locations and initialize the current index to zero and an invalid flag as False. The invalid flag becomes the flag to be checked if an invalid movement occurs or not. Simply if the invalid flag is True, we find any invalid movement. Conversely, when the invalid flag is False, we do not find any invalid movement. If it is True, then we check the any possible invalid movement using function isValid, as explained in Algorithm 4. If the procedure isValid returns True, then we can forward the index of the previous and current semantic locations. Conversely, if the function isValid returns False, we can set the invalid flag into True and store the current index i and the next index $i+1$ into new indexes *prev* and *curr*, respectively. When the invalid flag becomes True, then we move to another condition to resolve the invalid movement. However, we first need to check if index $i+1$ is at the end of a set. If it is at the end, we directly assign the semantic location at $i+1$ as the HMM output from the function getHMMCaseTwo depicted in Algorithm 6 which is applied for this case. In contrast, if the invalid flag is True, we check three conditions to validate the trajectory. First, if the semantic location in index $i+1$ is the same as that in index *curr*, then we assign the index *curr* as $i+1$. We execute this recurrently until we find the movement from the semantic location at index *curr* to different semantic locations at index $i+1$. Then, we again check if index $i+1$ is at the end of trajectory; then we get the HMM output from the function getHMMCaseTwo. Next, we set the semantic location from index $prev+1$ until $i+1$ as the output of HMM. Second, when we find different semantic locations at index $i+1$, we check the validity of a direct movement between semantic locations at index *prev* and at index $i+1$. If the direct movement is valid, then we obtain the HMM output from the function getHMMCaseOne applied for the case where index $i+1$ is not at the end of a trajectory, as described in Algorithm 5. Furthermore, we assign the semantic location from index $prev+1$ until *curr* as the output of HMM. The process of assigning the output of the rule-based HMM involves the process of validating the invalid semantic locations using the HMM. Again, we set the flag as False and index *prev* and *curr* as zero. Third, when we find different semantic locations at index $i+1$ but the direct movement between the semantic locations at index *prev* and at index $i+1$ is invalid, we assign the HMM output from function getHMMCaseTwo to the semantic locations from index $prev+1$ to index *curr*. Finally, we set the index *prev* as i and index *curr* as $i+1$. We perform this procedure iteratively until index i reaches at the end of the trajectory.

Algorithm 3: Validate the Extracted Trajectory

Input : - $\hat{S} = \{\hat{s}_1, \ldots, \hat{s}_i\}$, a set of estimated locations
Output: $\hat{S} = \{\hat{s}_1, \ldots, \hat{s}_i\}$, a valid set of estimated locations

```
Function getValidLoc(Ŝ)
    curr ← 0
    invalidflag ← false
    npred ← length of Ŝ
    for i in range(npred) do
        if i + 1 < npred then
            // check if an invalid movement occurs
            if not invalidflag then
                if isValid(Ŝ[i], Ŝ[i + 1]) then
                    continue
                else
                    invalidflag ← true, prev ← i, curr ← i + 1
                    // if index is at the end of array
                    if i + 1 = npred − 1 then
                        Assign a semantic location at index i + 1 as output of getHMMCaseTwo
                    end if
                else
                    if Ŝ[i + 1] = Ŝ[curr] then
                        curr ← i + 1
                        if i + 1 = npred − 1 then
                            Assign semantic locations at index prev + 1 until curr as output of getHMMCaseTwo
                        else if isValid(Ŝ[prev], Ŝ[i + 1]) then
                            Assign semantic locations at index prev + 1 until curr as output of getHMMCaseOne
                            invalidflag ← false, prev ← 0, curr ← 0
                        else
                            Assign semantic locations at index prev + 1 until curr as output of getHMMCaseTwo
                            prev ← i, curr ← i + 1
                        end if
                    end if
    end for
    return Ŝ
```

Algorithm 4: Checking the Validity of the Estimated Locations

Input : - $\hat{s}_{(prev)}$, previously valid estimated semantic location
- $\hat{s}_{(curr)}$, currently estimated semantic location
Output: true or false

```
Function isValid(ŝ_(prev), ŝ_(curr))
    // if current location is belonging
       to valid neighbor of previous
       location
    if (ŝ_(prev) and ŝ_(curr) ∈ VN((ŝ_(prev)))) then
        return true
    else
        return False
    end if
```

Algorithm 4 aims to check the validity of the movement between two consecutive semantic locations. If we are given a semantic location at index *prev* and a semantic location at index *curr* which belongs to the valid neighbor *VN* of a semantic location at index *prev*, then it returns True. This means that the movement from the semantic location at index *prev* to the next semantic location at index *curr* is considered to be valid. Otherwise, it will return False.

Additionally, we would like to emphasize that there are two important cases when applying the rule-based HMM. Algorithm 5 explains the procedure for obtaining a valid semantic location by utilizing the rule-based HMM for case 1. In case 1, the index $curr+1$ is not at the end of the trajectory, and direct movement from the semantic location at index *prev* to the semantic location at index $curr+1$ is considered valid. Initially, we already set the rules for the state transition probabilities where the invalid state transition has zero probability. This rule already guarantees that the GaussianHMM will output a valid state transition. However, we would like to create another empirical valid assumption for maintaining the robustness and also validating the initial set of estimated semantic locations from the output of DSLSTM. We can assume empirically that we have consecutive movements, e.g. A $\rightarrow$ B $\rightarrow$ C, where A, B, or C is represented by one or more than one movement within the same semantic locations. For instance, A $\rightarrow$ B $\rightarrow$ C can be the consecutive movements such as 5 $\rightarrow$ 5 $\rightarrow$ 7 $\rightarrow$ 7 $\rightarrow$ 6, where A=5, B=7, C=6. If we move from A to B then to C and the move from A to B is considered invalid and moving from A to C directly is considered valid, then we can make an empirical assumption that the valid estimated movement may be (A $\rightarrow$ A $\rightarrow$ C) or (A $\rightarrow$ C $\rightarrow$ C). In case 1, A or C can be the candidate of a valid semantic location. This empirical assumption is adopted based on the empirical experiment study between the estimated movements and actual movements of the users which has the highest robustness. We use three inputs such as a set of previously valid estimated locations *VS*, previously estimated semantic location $\hat{s}_{(prev)}$, and currently estimated semantic location $\hat{s}_{(curr+1)}$. We initialize an array of initial probabilities with a size of the number of states. If the semantic location at index *prev* is the same as the semantic location at index $curr+1$, then we set the initial probability (prior) at index $\hat{s}_{(prev)}-1$ as 1. Otherwise, we set the initial probability at index $\hat{s}_{(prev)}-1$ and $\hat{s}_{(curr+1)}-1$ as 0.5. Here, we utilize GaussianHMM with the input prior and a set of previously valid estimated locations. The GaussianHMM outputs the current trained transition probability matrix from the last valid semantic location at index *prev* and the most likely next valid state given the previous state as the semantic location at index *prev*. In this case, we aim to use only the currently trained transition probability matrix. We feed the desired GaussianHMM output with the semantic location at index *prev* and at index $curr+1$ into our proposed function rule explained in Algorithm 7. We introduce this rule to conduct validation based on the valid empirical assumption for case 1. Subsequently, we obtain a valid semantic location as the output.

Algorithm 6 has the same goal and produces an output such as that obtained by Algorithm 5. In case 2, the index $curr+1$ is the end of the trajectory and direct movement from the semantic location at index *prev* to the semantic location at index $curr+1$ is considered invalid. Because the direct movement between the semantic location at index *prev* and the semantic location at index $curr+1$ is invalid, this means that the semantic location at index $curr+1$ can be ignored. Therefore, we will not consider the semantic location at index $curr+1$ for validation in this case, and we omit it as an input to this function. We set the initial probability (prior) at index $\hat{s}_{(prev-1)}-1$ as 1. Here, we also utilize GaussianHMM with the input prior and a set of previously valid estimated locations. In this case, we utilize the GaussianHMM output directly with the initial transition rule. The output that we use is the predicted next valid state or semantic location given the previous state as the semantic location at index *prev*. Finally, we obtain the valid semantic location as the output.

Furthermore, Algorithm 7 describes the procedure of creating a rule which can validate movements of the extracted trajectory based on empirical valid assumption for case 1. Suppose we have the current trained transition probability matrix *hmmtransmat*, previously estimated semantic location $\hat{s}_{(prev)}$, and currently estimated semantic location $\hat{s}_{(curr)}$. We can get the probability of the movement from $\hat{s}_{(prev)}$ to $\hat{s}_{(curr)}$ and from $\hat{s}_{(prev)}$ and $\hat{s}_{(prev)}$. Therefore, we can compare if the probability of moving from the previous semantic location to the current semantic location is higher than staying at the same semantic location; then, we can conclude that the highest possible valid semantic location given the previous semantic location is the current semantic location. Otherwise, the highest possible valid semantic location given the previous semantic location is the previous semantic location. Finally, this proposed rule which follows the empirical valid assumption can maintain good robustness and can guarantee the validity of the set of estimated semantic locations.

Algorithm 5: Applying the Rule-Based HMM for Case 1

Input : - $VS = \{\hat{s}_1, \ldots, \hat{s}_{prev}\}$, a set of previously valid estimated locations
- $\hat{s}_{(prev)}$, previously estimated semantic location
- $\hat{s}_{(curr+1)}$, currently estimated semantic location

Output: *out*, valid estimated semantic location

Function `getHMMCaseOne` (VS, $\hat{s}_{(prev)}$, $\hat{s}_{(curr)}$)
  `// Initialize prior probability with 0`
  π = zeros(n_states)
  `// if previous location is the same as current location`
  if $\hat{s}_{(prev)} = \hat{s}_{(curr)}$ **then**
    $\pi[\hat{s}_{prev} - 1] \leftarrow 1$
  else
    $\pi[\hat{s}_{prev} - 1] \leftarrow 0.5$
    $\pi[\hat{s}_{curr+1} - 1] \leftarrow 0.5$
  end if
  $hmmtransmat, hmmpred \leftarrow$ GaussianHMM(π, VS)
  $out \leftarrow$ rule($hmmtransmat$, $\hat{s}_{prev}$, $\hat{s}_{curr+1}$)
  return *out*

Algorithm 6: Applying the Rule-Based HMM for Case 2

Input : - $VS = \{\hat{s}_1, \ldots, \hat{s}_{prev}\}$, a set of previously valid estimated locations
- $\hat{s}_{(prev)}$, previously estimated semantic location

Output: $\hat{s}_{(curr)}$, valid estimated semantic location

Function `getHMMCaseTwo` (VS, $\hat{s}_{(prev)}$, $\hat{s}_{(curr)}$)
  `// Initialize prior probability with 0`
  π = zeros(n_states)
  $\pi[\hat{s}_{prev} - 1] \leftarrow 1$
  $hmmtransmat, hmmpred \leftarrow$ GaussianHMM(π, VS)
  $out \leftarrow hmmpred$
  return *out*

Example 6: Figure 10 presents the procedure of applying rule-based refinement case 1 *on a set of semantic locations which consists of three steps. The set of semantic locations and rules here are devised based on the example of an indoor environment, as shown in Figure 2. Suppose we have a set of estimated semantic locations such as* $5 \rightarrow 6 \rightarrow 6 \rightarrow 6 \rightarrow 2 \rightarrow 2 \rightarrow 5 \rightarrow 5 \rightarrow 7 \rightarrow 8$. *In step A, we start checking the validity of movement of two consecutive movements from the first and second semantic locations. Then, we finally find the invalid movement from location* 6 *at index i to location* 2 *at index* $i + 1$. *In step B, we continue forwarding index i and* $i + 1$ *until we find different semantic locations at index* $i + 1$. *Now, the current index curr is set as index i, and we know that invalid movements exist from location* 6 *at index prev to location* 5 *at index curr* $+ 1$ *such as* $6 \rightarrow 2 \rightarrow 2 \rightarrow 5$. *First, we check the validity of direct movement from location* 6 *at index prev to location* 5 *at index curr* $+ 1$ *being valid; we set the priors of location* 6 *and* 5 *on average. Based on the empirical valid assumption, this is case 1; thus, we utilize the rule-based HMM approach for case* 1 *which also applies the rule described in Algorithm 5 with the input prior and past valid movements which can learn the past valid movements and output the valid estimated semantic location. Therefore, the candidates of a valid location are location* 5 *or location* 6. *In the next step C, the rule-based HMM resolves the invalid location* 2 *by providing a valid estimated location* 6. *Now, we have the valid movements from the start of the trajectory until index curr* $+ 1$ *such as* $5 \rightarrow 6 \rightarrow 6 \rightarrow 6 \rightarrow 6 \rightarrow 6 \rightarrow 5$.

Algorithm 7: Creating the Rule for Case 1

Input : - $hmmtransmat$, the current trained transition probability matrix given $\hat{s}_{(prev)}$
- $\hat{s}_{(prev)}$, previously valid estimated semantic position
- $\hat{s}_{(curr)}$, currently estimated semantic position

Output: $\hat{s}_{(prev)}$ or $\hat{s}_{(curr)}$, valid predicted semantic position

Function `rule` ($hmmtransmat$, $\hat{s}_{(prev)}$, $\hat{s}_{(curr)}$)
  `// transition probability from` $\hat{s}_{(prev)}$ `to` $\hat{s}_{(curr)}$
  $probprevcur \leftarrow hmmtransmat[prev - 1][curr - 1]$
  `// transition probability from` $\hat{s}_{(prev)}$ `to` $\hat{s}_{(prev)}$
  $probprevprev \leftarrow hmmtransmat[prev - 1][curr - 1]$
  if $probprevcur > probprevprev$ **then**
    `// output the current semantic location as valid location`
    return $\hat{s}_{(curr)}$
  else
    `// output the previous semantic location as valid location`
    return $\hat{s}_{(prev)}$
  end if

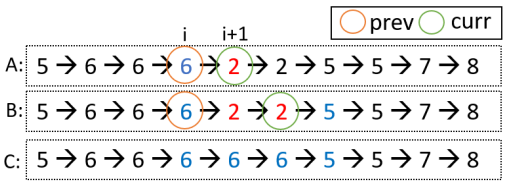

FIGURE 10. Rule-based refinement procedure for case 1.

Example 7: Figure 11 depicts the procedure of applying rule-based refinement case 2 *on a set of semantic locations which consists of three steps. We again find the invalid movement from location* 5 *at index i to location* 7 *at index* $i + 1$. *Then, we set the index prev as the index i and the index curr as the index* $i+1$ *at step A. Case 2 is when the validity of the direct movement from location* 5 *at index prev to location* 8 *at index curr* $+ 1$ *is considered invalid or at the end of a trajectory. We perform the same procedure as before by forwarding the index i and* $i+1$ *until we find a different semantic*

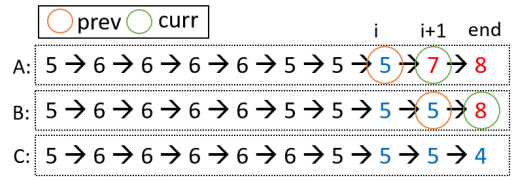

FIGURE 11. Rule-based refinement procedure for case 2.

location at index $i+1$. Therefore, we apply the rule-based HMM approach for case 2 by referring to Algorithm 6 to estimate a valid semantic location; however, we set a different prior setting. We set the prior equal to 1 for the only semantic location at index prev at step F. Then, we feed the input prior and past valid set of semantic locations into the rule-based HMM for case 2; thus, it can resolve the invalid locations with the valid ones. Because the empirical valid assumption cannot be satisfied in case 2, we let the GaussianHMM to estimate the valid semantic location directly. The invalid location is resolved by the proposed approach and outputs the valid location 5. At step B, we also forward the index so that index prev is assigned as index i and index curr is set as index $i+1$. Using the same validation procedure for case 2, we solve the invalid location at index curr with the prior setting based on the valid previous semantic location at index prev. Finally, we obtain the valid set of estimated semantic locations representing a semantic trajectory from the start to the end such as $5 \rightarrow 6 \rightarrow 6 \rightarrow 6 \rightarrow 6 \rightarrow 6 \rightarrow 5 \rightarrow 5 \rightarrow 5 \rightarrow 4$.

E. SEMANTIC TRAJECTORY EXTRACTION

Finally, given a valid set of estimated semantic locations $\hat{S} = \{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_i\}$, we can extract a valid semantic trajectory by concatenating a set of valid estimated semantic locations $\hat{S}$ with its corresponding time $\{\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_i\}$ from the start until the end of trajectory $T = 1$. We denote the valid extracted semantic trajectory as $\hat{ST} = \{(t_1, \hat{s}_1), (t_2, \hat{s}_2), \ldots, (t_i, \hat{s}_i)\}$. While the semantic trajectory extraction can be performed continuously, users can see their valid extracted semantic trajectories while they are still walking around in the building.

V. EXPERIMENTAL RESULTS

In this section, we detail the implementation and performance evaluation of our system.

A. EXPERIMENT SETUP

1) HARDWARE AND ENVIRONMENT SETUP

To evaluate the proposed system, we used a server machine equipped with an Intel®Core i7-8700 @ 3.20 GHz processor, with 32GB memory, NVIDIA GeForce GTX 1080 Ti 12GB, and running a 64-bit Ubuntu 18.04 operating system.

We implement our approach in Python 3.6 using Tensorflow-GPU 1.10.1 and Keras 2.2.2 performed with the CUDA Toolkit 9.0.176 and cuDNN 7.0.

2) DATASET

We used two real datasets for the experiments. The first dataset, D1, includes the RSSI observations collected directly in our building—Natural Science Research Building 4th Floor, Pusan National University. The data were collected using an Android smartphone for approximately three days over a duration of approximately 3.5 hours each day. The dataset contained two sub datasets—labeled training set and unlabeled testing set. The dataset contained a total of 215,806 samples. The labeled dataset contained semantic locations, RSSI readings from 10 deployed BLE beacons, and timestamps. RSSI measurements are represented as negative integer values, in which a higher value denotes higher proximity between the observed location and a given BLE beacon. Undetected beacon signals are indicated by a 0 RSSI value. The indoor environment has seven designated semantic locations. The floor map representation is illustrated in Figure 12.

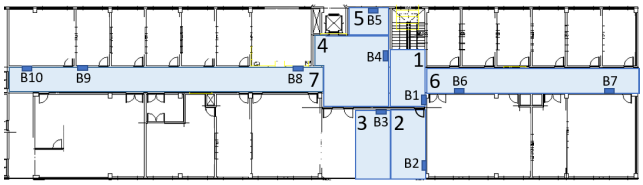

FIGURE 12. Floor map of the first dataset.

Additionally, we utilized a second publicly available real dataset D2 [27]. This dataset was collected using an iPhone 6S from 13 deployed beacons on the first floor of Waldo Library, Western Michigan University. This original dataset comprises two sub datasets—one labeled and one unlabeled—and it has total of 1,420 samples. The undetected RSSI readings are expressed as -200. The labeled dataset contains locations, timestamps, and RSSI readings. However, because the dataset is considerably small for experiments, we slightly modified the size by duplicating the samples to 2,758 samples. Additionally, the locations used in this dataset are expressed using a combination of a letter that represents a column and a number that depicts a row of the positions. Our approach utilizes the semantic location that is different from the locations provided in the dataset. Therefore, we also modified the provided locations into the semantic locations to support the purpose of our study. We updated the provided floor map with our designated semantic locations in the map. The modified floor map is displayed in Figure 13. The descriptions for those two datasets are summarized in Table 2.

The number of semantic locations and their neighbors may vary by different indoor dataset and environments such as a large museum or different buildings that contains multiple floors. In our experiment, the two datasets have different area, number of semantic locations, and neighboring relationship. Our proposed system performs well in such setting and is easily extensible to a larger area with different settings. Well defined semantic locations can be associated with point of interests (exhibition items, rooms type, specific areas). On the other hand, the neighboring semantic locations follows the restrictions of the indoor obstructions and connectivity such as walls, stairs, and corridors.

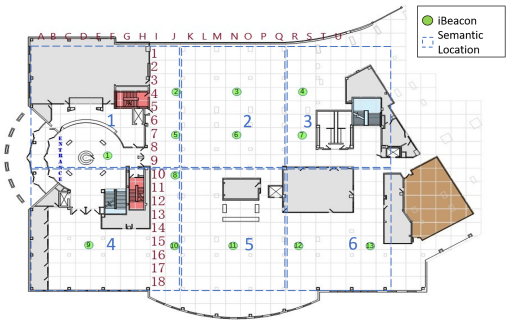

FIGURE 13. Modified floor map of the second dataset [27].

TABLE 2. Detailed dataset description.

Dataset	D1	D2
Deployed beacons	10	13
Semantic positions	7	6
Area	64.5 m × 17.4 m	80 m × 50 m
Training set	113295	1065
Test set	12588	355
Total samples	125883	1420

3) PARAMETER SETUP

Our approach requires parameters for the tumbling normalization window based on the number of samples and the number of past trajectories each input fingerprint set at time t wants to consider from time t until time $t - lookback + 1$. We decide the default value for the tumbling normalization window, and the lookback parameters depend on the dataset. The default window was 5000 for D1 and 500 for D2, whereas the default lookback was the same, 16, for both datasets. Then, we compared our proposed approach DAE-LSTM with rule-based refinement (DSLSTM+RR) with the related deep-learning based approaches used for indoor localization. We utilized the basic vanilla LSTM (VLSTM), SLTSM [21], DNN [18], and the proposed DAE-LSTM, which includes DAE and SLTSM without rule-based refinement (DSLSTM), as our baseline deep-learning based approaches because these deep learning-based approaches show comparable and promising performance. We also differ our approach with other LSTM approaches by set the dropout rate to 0.001. However, these approaches have not been implemented in a real-time manner yet. Therefore, we adopted these approaches for continuous trajectory extraction. We set the baseline approaches to our setting of semantic locations based on the experimental datasets. Table 3 indicates the parameter setting for each dataset in our experiments. We studied all approaches based on various parameters.

B. EVALUATION METRICS

We measured the performance of each approach through two metrics: robustness and validity. We applied these two metrics to an extracted semantic trajectory $\widehat{ST}$ and its related ground truth ST, where $ST = \{(t_i, s_i)\}$ and $\widehat{ST} = \{(t_i, \widehat{s_i})\}$.

TABLE 3. Parameter setup.

Dataset	Parameter Setup	
	Window Interval l (ms)	Lookback lb
D1	**5000**, 10000, 15000, 20000	8, 12, 16, **20**
D2	450, **500**, 550, 600	8, 12, 16, **20**

1) ROBUSTNESS

The robustness of the deep learning models (LSTM in this case) is represented by how the models can accurately estimate the semantic locations and the semantic trajectories. Therefore, we compared the estimated semantic trajectory $\widehat{ST}$ and the respective ground truth ST with the same length T. The formula of accuracy is described in Equation 19. The value of accuracy acc is within 0 to 1, where 0 means that the estimated trajectory is completely mispredicted and 1 means that all estimated semantic locations of a single semantic trajectory are correctly predicted.

$$acc(\widehat{ST}, S\,T) = \frac{\sum_{i=1}^{T} |\widehat{s_i} \cap s_i|}{|T|} \tag{19}$$

2) VALIDITY

The measure for validity aims to evaluate whether the extracted trajectory is valid or invalid. Here, we only evaluated the validity on the estimated semantic trajectory ST because the actual semantic trajectory $\hat{ST}$ is already a valid semantic trajectory. An invalid trajectory in one in which consecutive movement occurs from one semantic location to another semantic location that is not its valid neighbor. In other words, the distance from that location to another location is very large. An invalid movement violates the constraints of the indoor environment, such as movement through walls. Therefore, we can describe the validity loss $VL(\widehat{ST})$ using Equation 20,

$$VL(\widehat{ST}) = \sum_{i=1}^{T} ic(\hat{ST}) \tag{20}$$

where ic is the invalid cost, which indicates the cost when the invalid movement occurs. Assume a consecutive movement from $\hat{s_i}$ to $\hat{s_{i+1}}$. The invalid cost is 1 when an invalid movement occurs such as $\hat{s_{i+1}} \notin VN(\hat{s_i})$, whereas the invalid cost is 0 when no invalid movement occurs. The computation of the invalid cost ic is based on the cost computation used for the edit distance [1]. The maximum range of VL is equal to the maximum number of movements of an extracted trajectory $T - 1$. If VL is equal to $T - 1$, invalid movements occur in an entirely extracted trajectory. In contrast, if VL is 0, no invalid movement occurs in the entire trajectory. Therefore, we investigated the robust performance of each indoor localization approach based the designated parameters. Then, we evaluated the validity of the extracted trajectory for each approach.

C. ROBUSTNESS

1) VARYING WINDOW INTERVAL

We compared the performance of the proposed approach with those of the baseline approaches. Here, we set the lookback value to 16 for the inputs for the LSTM-based models including our proposed DAE-LSTM and rule-based refinement indicated by the DSLSTM+RR model. We investigated the results of robustness produced by our proposed approach and baseline approaches when the window interval is varied.

Figure 14 depicts the robustness of our proposed approach based on the accuracy metric of datasets D1 and D2. Our results show that the proposed approach shows a slight decline but can be regarded to show stable performance roughly when the subsequence tumbling window interval is increased in Dataset D1. For this evaluation, we set the window length to a considerably large value because our deep learning-based approaches can handle a large estimation within an acceptable time. Additionally, if the tumbling window is set to a small interval, it does not provide the general view of the performance of the deep learning models. The VLSTM and SLSTSM exhibited considerably good performance after our proposed approach. Thus, the temporal dependency of the extracted semantic trajectory is successfully learned and contributes to the performance of LSTM-based models. Our proposed DSLSTM shows almost the same robustness as the proposed DSLSTM which implements rule-based refinement (DSLSTM+RR). Table 4 indicates the obvious slight differences in robustness between the DSLSTM and our proposed approach DSLSTM+RR. Meanwhile, the performance of the DNN model was lower than that of the LSTM-based model because it does not consider the temporal dependency of the semantic trajectory.

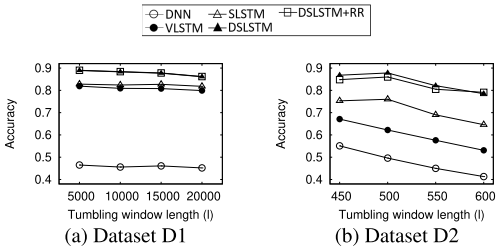

FIGURE 14. Varying the window interval.

In contrast, our robust DAE+SLSTM without rule-based refinement (DSLSTM) and the proposed approach with rule-based refinement (DSLSTM+RR) can produce comparable performances for dataset D2. We set a different window interval range compared to D1 because of the different size of D2. For this dataset, the DAE+SLTSM and the proposed approach exhibited the same trend as that for dataset D1, as shown in Table 5. Overall, we show that the robustness of our proposed approach is comparable to that the DSLSTM, and the proposed method outperforms other baseline approaches when applied on continuous inputs.

TABLE 4. Varying window interval (D1).

l	DSLSTM	DSLSTM+RR
5000	0.889	0.890
10000	0.883	0.884
15000	0.877	0.878
20000	0.861	0.862

TABLE 5. Varying window interval (D2).

l	DSLSTM	DSLSTM+RR
450	0.868	0.848
500	0.878	0.860
550	0.821	0.805
600	0.785	0.791

2) VARYING THE LOOKBACK

Next, we analyzed how the lookback parameter affects the robustness of our proposed approach and baseline approaches. In this case, we applied all approaches except the DNN to the full trajectory. The DNN was omitted because it does not consider the past trajectory of each fingerprint as input.

Figure 15 indicates the superior robustness of our approach and baseline approaches when the lookback is varied. It shows that when we consider the past trajectory longer, the robustness improves. Our proposed approaches (DSLSTM and DSLSTM+RR) indicate the most robust performance compared to other approaches. Our proposed approaches can produce the best robustness when we set a higher lookback. Additionally, importantly, implementing rule-based refinement (RR) can slightly improve the robustness. We present the difference in performance between the proposed DSLSTM and proposed DSLSTM+RR approaches for both datasets D1 and D2 in Table 6 and Table 7 respectively.

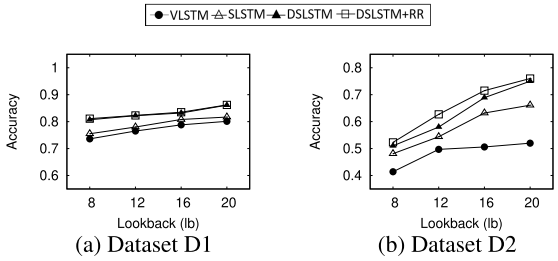

FIGURE 15. Varying the lookback.

D. VALIDITY

Because our proposed approach aims to ensure validity of the extracted semantic trajectory, we compared the validities of all approaches described in Table 8. We applied the default lookback value of 20 in this experiment. Here, we aimed to show how our proposed rule-based refinement (RR) can be applied to another deep learning-based approach such as DNN [18]; therefore, we included one additional approach, DNN+RR (DNN with our proposed rule based refinement), in the validity evaluation. As expected, our proposed approach (DSLSTM+RR) could extract valid semantic

TABLE 6. Varying lookback (D1).

lb	DSLSTM	DSLSTM+RR
8	0.807	0.811
12	0.822	0.823
16	0.832	0.835
20	0.861	0.862

TABLE 7. Varying lookback (D2).

lb	DSLSTM	DSLSTM+RR
8	0.510	0.523
12	0.580	0.627
16	0.689	0.715
20	0.750	0.760

TABLE 8. Validity loss of each method on each dataset.

Approach	Valid Loss VL	
	D1	D2
DNN	10674	13
DNN+RR	0	0
VSLTM	47	11
SLSTM	39	8
DSLSTM	29	6
DSLSTM+RR	0	0

trajectories with a zero valid loss *VL*. The DNN+RR approach returns the maximum validity with $VL = 0$; this means that our proposed rule-based refinement is extensible to other deep learning-based approaches. However, surprisingly, the proposed DSLSTM can achieve good validity with the valid loss is only approximately 29 and 6 for datasets D1 and D2, respectively. This means that very few invalid movements occur in the extracted semantic trajectory of D1 which has the maximum length. Moreover, the DSLSTM can also extract valid semantic trajectories for dataset D2. In this case, we can conclude that the better robustness of the deep learning-based approach is responsible for the better validity of the extracted semantic trajectory.

E. DISCUSSION

The experimental results indicate that our proposed approaches that utilize the combined DAE-LSTM model with or without rule based refinement (DSLSTM and DSLSTM+RR) can estimate the robust semantic locations and extract valid semantic trajectories in different indoor settings whereas basic LSTM and stacked LSTM cannot. Compared with various LSTM approaches, the DSLSTM with rule refinement works better than the basic vanilla LSTM and the stacked LSTM because of its denoising architecture. The basic vanilla LSTM and stacked LSTM is more vulnerable to overfit the noisy data thus extracts invalid movements. In addition, our DSLSTM with rule refinement extracts semantic trajectories with zero invalid movement. However, maintaining the validity of the extracted semantic trajectory can sometimes affect the robustness of the approach. For instance, when we set the tumbling window interval to 450, 500, or 550, our proposed approach DSLSTM+RR produces lower robustness than the proposed DAE-LSTM without a rule-based refinement (DSLSTM) for dataset D2. Furthermore, the higher robustness of the deep learning-based approaches encourages the validity of the extracted semantic trajectory.

VI. CONCLUSION

In this study, we propose a combined deep learning based approach called DSLSTM with RR to perform robust and valid semantic trajectory extraction. First, we built and trained the proposed DSLSTM model to obtain a robust model that can estimate the semantic locations accurately. Second, we introduced a rule-based refinement that applies a rule-based HMM to extract valid semantic trajectories. Rule-based refinement utilizes the set of rules which can be applied with the HMM to produce valid movements of an extracted semantic trajectory. The experimental results demonstrate that our proposed approach can extract robust and valid semantic trajectories. Compared with other state-of-the-art approaches, our proposed approach presents comparable robustness of indoor localization. The experimental results also show that the proposed rule-based refinement can be extensible to other deep learning-based approaches for valid semantic trajectory extraction.

In the future, we aim to extend this work for other location-based services such as next-to visit recommendation, activity recognition, and users' behavior discovery. Another interesting future direction is applying our proposed approach on image or video-based datasets for indoor localization.

REFERENCES

[1] A. I. Baba, M. Jaeger, H. Lu, T. B. Pedersen, W.-S. Ku, and X. Xie, "Learning-based cleansing for indoor RFID data," in *Proc. Int. Conf. Manage. Data*, Jun. 2016, pp. 925–936.

[2] X. Wang, Z. Yu, and S. Mao, "DeepML: Deep LSTM for indoor localization with smartphone magnetic and light sensors," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[3] R. Algabri and M.-T. Choi, "Deep-Learning-Based indoor human following of mobile robot using color feature," *Sensors*, vol. 20, no. 9, p. 2699, May 2020, doi: 10.3390/s20092699.

[4] Y. Wu, P. Chen, F. Gu, X. Zheng, and J. Shang, "*HTrack*: An efficient heading-aided map matching for indoor localization and tracking," *IEEE Sensors J.*, vol. 19, no. 8, pp. 3100–3110, Apr. 2019.

[5] L. Zhang, Z. Chen, W. Cui, B. Li, C. Chen, Z. Cao, and K. Gao, "WiFi-based indoor robot positioning using deep fuzzy forests," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 10773–10781, Nov. 2020, doi: 10.1109/JIOT.2020.2986685.

[6] F. Danış and A. Cemgil, "Model-based localization and tracking using Bluetooth low-energy beacons," *Sensors*, vol. 17, no. 11, p. 2484, Oct. 2017.

[7] H. Ramadhan, Y. Yustiawan, and J. Kwon, "Applying movement constraints to BLE RSSI-based indoor positioning for extracting valid semantic trajectories," *Sensors*, vol. 20, no. 2, p. 527, Jan. 2020, doi: 10.3390/s20020527.

[8] D. Yamamoto, R. Tanaka, S. Kajioka, H. Matsuo, and N. Takahashi, "Global map matching using BLE beacons for indoor route and stay estimation," in *Proc. 26th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2018, pp. 309–318.

[9] F. J. Aranda, F. Parralejo, F. J. Álvarez, and J. Torres-Sospedra, "Multi-slot BLE raw database for accurate positioning in mixed indoor/outdoor environments," *Data*, vol. 5, no. 3, p. 67, Jul. 2020, doi: 10.3390/data5030067.

[10] F. Piccialli, F. Giampaolo, G. Casolla, V. S. D. Cola, and K. Li, "A deep learning approach for path prediction in a location-based IoT system," *Pervas. Mobile Comput.*, vol. 66, Jul. 2020, Art. no. 101210, doi: 10.1016/j.pmcj.2020.101210.

[11] Y. Xu, M. Zhou, W. Meng, and L. Ma, "Optimal KNN positioning algorithm via theoretical accuracy criterion in WLAN indoor environment," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2010, pp. 1–5.

[12] Y. Peng, W. Fan, X. Dong, and X. Zhang, "An iterative weighted KNN (IW-KNN) based indoor localization method in Bluetooth low energy (BLE) environment," in *Proc. Intl IEEE Conf. Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart World Congr. (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Jul. 2016, pp. 794–800.
[13] Q. Wang, R. Sun, X. Zhang, Y. Sun, and X. Lu, "Bluetooth positioning based on weighted *K*-nearest neighbors and adaptive bandwidth mean shift," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 5, May 2017, Art. no. 155014771770668. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1550147717706681
[14] C.-L. Wu, L.-C. Fu, and F.-L. Lian, "WLAN location determination in e-home via support vector classification," in *Proc. IEEE Int. Conf. Netw., Sens. Control*, vol. 2, Mar. 2004, pp. 1026–1031.
[15] A. Chriki, H. Touati, and H. Snoussi, "SVM-based indoor localization in wireless sensor networks," in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2017, pp. 1144–1149.
[16] Y. Rezgui, L. Pei, X. Chen, F. Wen, and C. Han, "An efficient normalized rank based SVM for room level indoor WiFi localization with diverse devices," *Mobile Inf. Syst.*, vol. 2017, Jul. 2017, Art. no. 6268797.
[17] J. Liu, N. Liu, Z. Pan, and X. You, "AutLoc: Deep autoencoder for indoor localization with RSS fingerprinting," in *Proc. 10th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2018, pp. 1–6.
[18] K. S. Kim, S. Lee, and K. Huang, "A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on Wi-Fi fingerprinting," *Big Data Anal.*, vol. 3, no. 1, p. 4, Dec. 2018.
[19] M. Yan, F. Xu, S. Bai, and Q. Wan, "A noise reduction fingerprint feature for indoor localization," in *Proc. 10th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2018, pp. 1–6.
[20] P. Zhang, W. Ouyang, P. Zhang, J. Xue, and N. Zheng, "SR-LSTM: State refinement for LSTM towards pedestrian trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12085–12094.
[21] A. Sahar and D. Han, "An LSTM-based indoor positioning method using Wi-Fi signals," in *Proc. 2nd Int. Conf. Vis., Image Signal Process.*, Aug. 2018, p. 43.
[22] P. Dai, Y. Yang, M. Wang, and R. Yan, "Combination of DNN and improved KNN for indoor location fingerprinting," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–9, Mar. 2019.
[23] C. Xiao, D. Yang, Z. Chen, and G. Tan, "3-D BLE indoor localization based on denoising autoencoder," *IEEE Access*, vol. 5, pp. 12751–12760, 2017.
[24] X. Li, M. Li, Y.-J. Gong, X.-L. Zhang, and J. Yin, "T-Des.: Destination prediction based on big trajectory data," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2344–2354, Aug. 2016.
[25] P. Viaene, A. Vanclooster, K. Ooms, and P. De Maeyer, "Thinking aloud in search of landmark characteristics in an indoor environment," in *Proc. Ubiquitous Positioning Indoor Navigat. Location Based Service (UPINLBS)*, Nov. 2014, pp. 103–110.
[26] A. Karatzoglou, A. Jablonski, and M. Beigl, "A Seq2Seq learning approach for modeling semantic trajectories and predicting the next location," in *Proc. 26th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2018, pp. 528–531.
[27] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J. S. Oh, "Semisupervised deep reinforcement learning in support of IoT and smart city services," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 624–635, Apr. 2018.
[28] L. Kanaris, A. Kokkinis, A. Liotta, and S. Stavrou, "Fusing Bluetooth beacon data with Wi-Fi radiomaps for improved indoor localization," *Sensors*, vol. 17, no. 4, p. 812, Apr. 2017.
[29] C. Li, Z. Qiu, and C. Liu, "An improved weighted K-nearest neighbor algorithm for indoor positioning," *Wireless Pers. Commun.*, vol. 96, no. 2, pp. 2239–2251, Sep. 2017.
[30] W. Xue, X. Hua, Q. Li, W. Qiu, and X. Peng, "Improved clustering algorithm of neighboring reference points based on KNN for indoor localization," in *Proc. Ubiquitous Positioning, Indoor Navigat. Location-Based Services (UPINLBS)*, Mar. 2018, pp. 1–4.
[31] Z. Ezzati Khatab, V. Moghtadaiee, and S. A. Ghorashi, "A fingerprint-based technique for indoor localization using fuzzy least squares support vector machine," in *Proc. Iranian Conf. Electr. Eng. (ICEE)*, May 2017, pp. 1944–1949.
[32] X. Guo, N. Ansari, L. Li, and H. Li, "Indoor localization by fusing a group of fingerprints based on random forests," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4686–4698, Dec. 2018.
[33] S. Lee, J. Kim, and N. Moon, "Random forest and WiFi fingerprint-based indoor location recognition system using smart watch," *Hum.-centric Comput. Inf. Sci.*, vol. 9, no. 1, p. 6, Dec. 2019.
[34] D. Han, H. Rho, and S. Lim, "HMM-based indoor localization using smart Watches' BLE signals," in *Proc. IEEE 6th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2018, pp. 296–302.
[35] A. Adege, H.-P. Lin, G. Tarekegn, and S.-S. Jeng, "Applying deep neural network (DNN) for robust indoor localization in multi-building environment," *Appl. Sci.*, vol. 8, no. 7, p. 1062, Jun. 2018.
[36] W. Zhang, K. Liu, W. Zhang, Y. Zhang, and J. Gu, "Deep neural networks for wireless localization in indoor and outdoor environments," *Neurocomputing*, vol. 194, pp. 279–287, Jun. 2016.
[37] S. Aikawa, S. Yamamoto, and M. Morimoto, "WLAN finger print localization using deep learning," in *Proc. IEEE Asia–Pacific Conf. Antennas Propag. (APCAP)*, Aug. 2018, pp. 541–542.
[38] Z. E. Khatab, A. Hajihoseini, and S. A. Ghorashi, "A fingerprint method for indoor localization using autoencoder based deep extreme learning machine," *IEEE Sensors Lett.*, vol. 2, no. 1, pp. 1–4, Mar. 2018.
[39] H. Li, H. Lu, G. Chen, K. Chen, Q. Chen, and L. Shou, "Toward translating raw indoor positioning data into mobility semantics," *ACM/IMS Trans. Data Sci.*, vol. 1, no. 4, p. 26, 2020, doi: 10.1145/3385190.
[40] N. Mou, H. Wang, H. Zhang, and X. Fu, "Association rule mining method based on the similarity metric of tuple-relation in indoor environment," *IEEE Access*, vol. 8, pp. 52041–52051, 2020, doi: 10.1109/ACCESS.2020.2980952.
[41] H. Noureddine, C. Ray, and C. Claramunt, "Semantic trajectory modelling in indoor and outdoor spaces," in *Proc. 21st IEEE Int. Conf. Mobile Data Manage. (MDM)*, Versailles, France, Jun. 2020, pp. 131–136, doi: 10.1109/MDM48529.2020.00035.
[42] C. Zhang, J. Han, L. Shou, J. Lu, and T. La Porta, "Splitter: Mining fine-grained sequential patterns in semantic trajectories," *Proc. VLDB Endowment*, vol. 7, no. 9, pp. 769–780, 2014.

YOGA YUSTIAWAN received the Bachelor of Engineering degree from the Department of Informatics Engineering, Telkom University, Indonesia, in 2015, and the master's degree in engineering from the Department of Big Data, Pusan National University, South Korea, in 2019. He is currently working as a Data Scientist with CIMB Niaga Bank, Indonesia. His main research interests include big data analysis, scalable data mining for the Internet of Things, deep learning, and databases.

HANI RAMADHAN received the joint master's degree from the Informatics Engineering Department, Institut Teknologi Sepuluh Nopember, Indonesia and the Complex System and Interaction Department, Universite de Technologie de Compiegne, France. He is currently pursuing the Ph.D. degree from the Big Data Department, Pusan National University, South Korea. His research interests include data mining, machine learning, and computer vision.

JOONHO KWON received the B.S., M.S., and Ph.D. degrees from the School of Electrical Engineering and Computer Engineering, Seoul National University, South Korea, in 1999, 2001, and 2009, respectively. He is currently a Professor with the School of Computer Science and Engineering at Pusan National University, South Korea. His current research interests include big data management and analytics, NoSQL databases, the IoT data query processing, XML indexing and query processing, Web services, RFID data management, and serious games.