IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 October 2023, accepted 16 November 2023, date of publication 21 November 2023,
date of current version 29 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3335665

RESEARCH ARTICLE

Anomalous Indoor Human Trajectory Detection Based on the Transformer Encoder and Self-Organizing Map

DOI THI LAN AND SEOKHOON YOON, (Member, IEEE)
Department of Electrical, Electronic and Computer Engineering, University of Ulsan, Ulsan 44610, South Korea

Corresponding author: Seokhoon Yoon (seokhoonyoon@ulsan.ac.kr)

This work was supported in part by Institute of Information & communication Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2020-0-00869, Development of 5G-based Shipbuilding & Marine Smart Communication Platform and Convergence Service), and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2021R1I1A3051364.

ABSTRACT Anomalous human trajectory detection is a critical task in security surveillance in working areas. To identify anomalous human trajectories, understanding features of their movement plays an important role. Therefore, in this work, a Transformer encoder and self-organizing map-based model called TENSO is proposed to learn trajectory characteristics for detecting anomalies. In particular, the proposed model learns the internal characteristics of normal trajectories and clusters of normal trajectory representations in a latent space. To learn the internal characteristics of normal trajectories, the encoder of Transformer with a self-attention mechanism first encodes trajectories into sequences of embedding vectors of trajectory points in the latent space. Then, a decoder reconstructs the trajectories from the latent space. In addition, to learn clusters of normal trajectory representations in the latent space, the self-organizing map (SOM) layer is used, which gets its input as the output of the Transformer encoder. In the training phase, the TENSO model is trained using a total loss of trajectory reconstruction and SOM losses. In the anomaly detection phase, a test trajectory is evaluated to determine whether it is an anomaly based on trajectory reconstruction errors and the quantization error on the SOM. In this phase, a new metric is proposed which, namely WS, is the weighted sum of recall and precision to choose the appropriate threshold for detecting anomalies. The TENSO model-based framework is evaluated using two real trajectory datasets: MIT Badge and sCREEN. Experimental results show that the proposed framework identifies anomalies effectively and outperforms the baselines.

INDEX TERMS Anomalous trajectory detection, indoor human trajectory, transformer encoder, self-organizing map.

I. INTRODUCTION

Location-acquisition tools like the GPS, smartphone, and sensors have helped generate massive quantities of position data from moving objects. The diversity in trajectory data encourages a lot of research on trajectory data mining, such as human trajectory prediction [1], [2], [3], user route recommendation [4], [5], [6], [7], and trajectory clustering [8], [9], [10]. In recent years, anomalous trajectory detection has become an important research topic in many applications. For example, for taxi services, anomalous taxi trajectories are associated with problems like traffic jams, accidents, and taxi driver fraud. Thus, identifying uncommon taxi trips can enhance the quality of the service [11], [12], [13]. Besides, one essential aspect of security surveillance in public areas is anomaly event detection. By evaluating people's movements in public spaces, it is possible to identify abnormal events such as terrorism, violent attacks, and accidents [14], [15].

The associate editor coordinating the review of this manuscript and approving it for publication was Yiqi Liu.

However, the above studies mainly relate to anomalous trajectory detection in outdoor spaces.

With indoor areas, trajectory data have opened many new research directions. In particular, customers' trajectories in shops can be investigated to determine their purchasing habits [16]. This enables owners to optimize product positioning and shop design. Human location prediction systems for indoor environments play a significant role in location-based services [17], [18]. Additionally, anomalous human movements in indoor spaces often relate to serious situations, such as violent attacks, theft, and fire. Thus, detecting abnormal movements by people can improve safety in indoor environments. In this paper, our objective is to identify abnormal human trajectories in indoor spaces.

Anomaly detection methods in trajectory data can be divided into two main categories: traditional detection methods and deep learning-based detection methods. With traditional detection methods (e.g., distance-based, density-based, and clustering-based), detecting anomalies is mainly based on the relationship between the test trajectory and the remaining trajectories [13], [19], [20]. These methods do not discover internal characteristics and sequential information in the trajectories. Deep learning-based detection methods often focus on learning embedding vectors that capture spatial-temporal information in trajectories. Detecting anomalies is based on the embedding vectors [21], [22], [23], [24], [25]. However, these methods do not discover relationships between trajectory representations (e.g., distance, density, and clusters of trajectory representations) in latent space to use them for detecting abnormal trajectories.

From the above aspects, we aim at building a novel deep learning model that learns normal trajectories' interior features and clusters of normal trajectory representations in latent space to detect anomalies. Since the proposed model performs the roles of deep learning models (i.e., learning trajectory representations) and traditional methods (i.e., identifying trajectory clusters in a representation space), the effectiveness of the proposed framework in anomaly detection can be improved compared with existing methods.

In particular, a deep learning model called TENSO, which is based on the Transformer encoder and self-organizing map (SOM), is proposed in this paper. To capture internal characteristics of normal trajectories in a latent space, the Transformer encoder is first used. Each trajectory is encoded by an output sequence that captures the correlation between points in the trajectory through a self-attention mechanism. To learn more helpful information in latent space, the TENSO model discovers both positional information of trajectories' points and their indoor semantic information (e.g., working rooms, meeting rooms, and corridors). Then, a decoder reconstructs the trajectories from their representations in the latent space.

To learn the clusters of normal trajectory representations in the latent space, a SOM layer is used in this work. The SOM is a data clustering model containing interconnected neurons on a low-dimensional map [26]. Each neuron is represented by a prototype vector. Note that prototype vectors belong to the same space as the input data for the SOM. While training the SOM, the prototype vectors are updated and forwarded to input data. In TENSO, the SOM input data space contains latent representations of trajectories that are the output of the Transformer encoder. Besides, the number of prototype vectors is chosen much less than the number of trajectories in this work. Thus, if trajectory representations are close to each other in the latent space, they are represented by the same prototype vector on the SOM.

To train the TENSO model, a total loss of the trajectory reconstruction and SOM losses is used. After the model is trained, anomaly detection is performed when a new trajectory comes. In particular, a trajectory's anomaly score (AS) is determined using the TENSO model. If the AS exceeds a given threshold, the trajectory is detected as an anomaly. In this work, the anomaly threshold is determined based on trajectories' anomaly scores in the training set. A new factor, the weighted sum (WS) of precision and recall, is proposed to choose the threshold value.

Experiments for the proposed framework's evaluation are performed using two real datasets: MIT Badge and sCREEN. With the MIT Badge dataset, the best results from our framework are about 90% and 95% in terms of f1-score with hypothesized and synthesized anomalies, respectively. With the sCREEN dataset, the proposed framework also effectively detects synthesized anomalies, with the best performance of about 96% for the f1-score. In summary, this work's main contributions are as follows.

- A deep learning model that learns trajectory representations and their clusters in latent space is proposed. To learn trajectory representations, the Transformer encoder is used to encode trajectories based on the self-attention mechanism. In addition, the SOM layer learns typical trajectory representation groups in latent space. To train the proposed model, a total loss of trajectory reconstruction and SOM losses is proposed. In detecting anomalies, the abnormality of a trajectory is determined using trajectory reconstruction errors and the quantization error on the SOM. If the anomaly score of a trajectory exceeds a given threshold, the trajectory is marked as an anomaly.
- We propose a new method for determining an appropriate anomaly threshold in this work. The proposed method chooses the threshold based on the trajectories' anomaly scores in the training set. To find the proper threshold value, the weighted sum factor of recall and precision is proposed.
- The proposed anomalous indoor trajectory detection framework is evaluated using two real-world datasets. The results indicate that our approach performs better than existing approaches in anomalous trajectory detection.

This paper is organized as follows. We start in Section II by discussing the background and related works. The

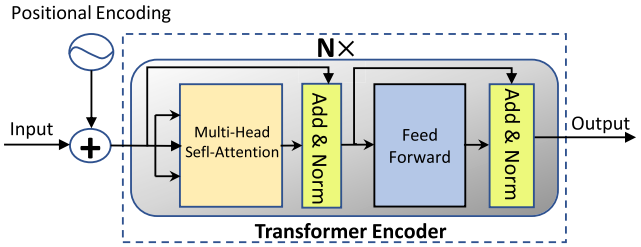

FIGURE 1. Transformer encoder.

anomaly detection problem is described in Section III. Section IV provides a detailed introduction to the proposed methodology. Algorithm performance is assessed in Section V. Section VI presents the paper's conclusions.

II. BACKGROUND AND RELATED WORKS

This section begins by presenting the background of the Transformer encoder and SOM. Then, we discuss studies on anomalous trajectory detection.

A. TRANSFORMER ENCODER

Transformer is a network architecture based on an attention mechanism and consists of two main parts: an encoder and a decoder [27]. This neural network can effectively replace recurrent neural network (RNN) and convolutional neural network (CNN) architectures in sequential tasks. In this work, we only consider the Transformer encoder, which is presented in Figure 1. It contains two sub-layers: a layer of multi-head self-attention and a position-wise fully connected feed-forward network.

- **Multi-head self-attention.** A single attention layer called Scaled Dot-Product Attention is the main calculation layer in multi-head attention. It consist of three inputs: queries, keys of dimension d_k, and values of dimension d_v. To determine the weights of the values, they compute the dot products of the query with each key, divide each result by $\sqrt{d_k}$, and then apply a softmax function. The queries, keys, and values are packed as separate matrices: Q, K, and V. The equation for a single attention is

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

  Note that in the Transformer encoder, the keys, queries, and values are obtained from the same place, and the attention is known as self-attention. Since performing parallel self-attention layers on multiple subspaces of the input sequence is better than on a single space, the input sequence is projected h times with learned linear projections before applying self-attention layers. This mechanism is known as multi-head self-attention in the Transformer encoder and is performed as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \quad (2)$$

  where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$, and the linear projections are matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

- **Position-wise feed-forward network.** This sub-layer is applied to each position of the output sequence of multi-head self-attention. It consists of a dense layer with ReLU activation and a linear layer:

$$\text{FFN}(x) = W_2 \times \text{ReLU}(W_1 x + b_1) + b_2, \quad (3)$$

  where x is the input of the sub-layer, with W_1, W_2, b_1, and b_2 as parameters of the network.

The output from each of these two sub-layers of the Transformer encoder are normalized and added by an Add & Norm layer.

B. SELF-ORGANIZING MAP

The SOM is a competitive learning neural network known as a data clustering model [28], [29]. This network has two layers: an input layer, and an output layer of interconnected neurons (often called units or nodes) in a two-dimensional grip map.

Assume that a set of input data samples is defined $X = \{x_i\}_{1 \le i \le L}, x_i \in \mathbb{R}^D$. A SOM has V units, and each unit is represented by a corresponding prototype vector $\{m_v\}_{1 \le v \le V}$. Prototype vectors belong to the same space as input data (i.e., $\mathbb{R}^D$). In each iteration of SOM training, all prototype vectors are updated. The level of the update depends on the correlation between prototype vectors and the input data, which is determined by a distance metric. When an input data sample comes, the prototype vector with the smallest distance is determined. The unit on the map that corresponds to this prototype vector is called the best matching unit (BMU). A definition for BMU b_i of x_i is

$$b_i = \underset{v}{\text{argmin}}(\text{Dist}(x_i, m_v)), \quad (4)$$

where Dist(.) is the chosen distance metric.

On the grid map, the Manhattan distance $d(i, j)$ between two nodes i and j is determined. A temperature parameter, Tem, and a neighborhood function, $K^{\text{Tem}}(d)$, of SOM are also defined. Temperature parameter Tem at iteration iter is determined as follows:

$$\text{Tem}(\text{iter}) = \text{Tem}_{\max}(\frac{\text{Tem}_{\min}}{\text{Tem}_{\max}})^{\frac{\text{iter}}{\text{Num}_{\text{iters}}}}, \quad (5)$$

where $\text{Tem}_{\max}$ and $\text{Tem}_{\min}$ are the initial and final temperatures, respectively. $\text{Num}_{\text{iters}}$ is the number of iterations. The function $K^{\text{Tem}}(d)$ determines the neighborhood radius around a unit on the map, and can be defined as a Gaussian neighborhood function as follows:

$$K^{\text{Tem}}(d) = e^{-\frac{d^2}{\text{Tem}^2}}. \quad (6)$$

In its training phase, the SOM gets each input sample x_i and updates all prototype vectors by forwarding them to closer x_i. The weight for updating is the value of the neighborhood function around the BMU of x_i. Thus, close neighboring

units will be updated more than farther ones. The updating procedure is presented as follows:

$$m_v \leftarrow m_v + \alpha \times K^{\text{Tem}}(d(b_i, v))(x_i - m_v), \tag{7}$$

where α is the learning rate.

C. ABNORMAL TRAJECTORY DETECTION

Abnormal trajectory detection methods are mainly grouped into two categories: traditional methods and deep learning-based methods.

1) TRADITIONAL METHODS

With traditional detection methods, the characteristics of the trajectory (e.g., distance from other trajectories, density, and the relationship with clusters in the dataset) are discovered to detect anomalies.

The distance-based anomalous trajectory detection approach uses a distance function to determine similarities among trajectories. To identify an anomalous trajectory, a similarity measure is provided [20], [30]. Zhu et al. presented Time-dependent Popular Routes-based trajectory Outlier detection (TPRO) approach in [20]. The most common routes on each timestamp are utilized in this study to identify temporal anomalies. They split the trajectory dataset into various groups using a partitioning approach. A reference trajectory was used to represent each group. The edit distance between each group's reference trajectory and the most travelled routes was then determined. If the difference between the reference trajectory and the common routes was greater than a certain threshold, the trajectory group was marked as an anomaly group. Saleem et al. [30] introduced the Road segment Partitioning towards Anomalous Trajectory detection (RPAT) technique. The trajectories were split into sub-trajectories based on the road segments. The features utilized to determine the score for each sub-trajectory were the speed, flow rate, and visited time. The score of each trajectory was the sum of the scores of each sub-trajectory. An anomaly was shown if the trajectory score was larger than a user-specified threshold.

With density-based anomalous trajectory detection methods, the neighbor density of trajectories is computed to identify anomalies. By using a partition-and-detect approach, the authors in [19] proposed the TRAjectory Outlier Detection (TRAOD) algorithm to find sub-trajectory outliers. The three elements of a new distance function (i.e., perpendicular, parallel, and angle distances) are proposed. This distance metric determines how similar sub-trajectories are. Then, a distance threshold is utilized to calculate the density of sub-trajectories. If a sub-trajectory's density is less than a certain threshold, it is identified as an anomaly.

With clustering-based detection methods, a suitable clustering technique was first applied to find clusters in the dataset. Then, an anomalous trajectory was detected if it was not associated with any clusters. The authors in [31] proposed an abnormal trajectory detection approach using a hierarchical clustering algorithm. The longest common subsequence (LCSS) metric was first utilized to measure distance of trajectories. Then, the hierarchical clustering algorithm was used to determine clusters of trajectories. Finally, they labelled a new trajectory based on the relationship between this trajectory and clusters. In our previous work, a DBSCAN-based method for detecting abnormal trajectories was also proposed [32]. To improve the effectiveness of the distance metric for indoor human trajectories, we proposed a new metric called LCSS_IS, extended from LCSS. Besides, the Eps parameter of DBSCAN was determined using a novel DBSCAN cluster validation index (DCVI). In detecting anomalies, the clusters of normal trajectories in the dataset were first found using DBSCAN. Then, a new trajectory was marked as an anomaly if it did not belong to any clusters in the dataset.

From the above studies, we can see that the traditional detection methods mainly focus on the relationship between trajectories in the dataset to detect anomalies. However, these methods did not discover the trajectory's internal features and sequential information. In contrast, our work designs a deep learning model with the Transformer encoder to learn the correlation between points within a trajectory and its sequential nature. Besides, the proposed model also learns the clusters of trajectory representations based on the SOM layer.

2) DEEP LEARNING-BASED METHODS

With recent neural network development, anomalous trajectory detection methods have started to focus on learning trajectory representations.

In particular, RNNs, used for learning sequential data, are widely applied for outlier detection tasks in trajectory data. The authors in [21] proposed a deep learning model called Anomalous Trajectory Detection using Recurrent Neural Network (ATD-RNN) to identify abnormal trajectories. This model learned the trajectory embedding that kept normal trajectories' internal characteristics and their sequential information. ATD-RNN was trained by minimizing the cross-entropy loss function using a labeled dataset. With a new trajectory, the model predicted the probability of detecting the trajectory as an anomaly.

The study in [33] introduced an LSTM autoencoder-based Seq2Seq model to identify abnormal vehicle routes. An LSTM encoder mapped each input route into a vector, which captured the input route's characteristics. Then, an LSTM decoder reconstructed the encoded route. An anomaly was detected if it was not reconstructed correctly by the autoencoder. Similarly, a framework was proposed for detecting dangerous driving behavior and hazardous roads using autoencoders [22]. In particular, the autoencoders were trained to learn a latent space that captures the input sequences' most representative characteristics. The difference between the input and reconstructed sequences

was also used to indicate outliers. In the work, the authors studied autoencoders based on CNN and LSTM architectures for anomaly detection tasks.

In recent years, due to the occurrence of the Transformer architecture with an attention mechanism, capturing long-range dependence in sequential data is performed more effectively. Anomaly detection methods based on this neural network have also emerged gradually. For example, detecting anomalies in electrocardiogram (ECG) signals based on the Transformer was introduced in [34]. In this paper, the authors used the Transformer encoder to learn normal data patterns. Anomalies are detected using errors between the predicted and original ECG signals. The study in [23] also proposed an anomalous detection method based on the Transformer for improving safety and predicting risks in traffic. Their model used the Universal Transformer encoder to learn trajectories' embeddings by keeping information on trajectory points. Like the study [21], this model was also trained using the cross-entropy loss function, and a labeled dataset for abnormal and normal trajectories was required. Both above Transformer-based models learned to capture the interior features of normal samples in input data. However, they do not learn the relationship between representations of data in latent space to detect anomalies. Besides, the study in [34] does not provide an effective mechanism for choosing an appropriate anomaly threshold. This work still needs to set a specific value in the formulation of determining the anomaly threshold.

By contrast, in our proposed approach, learning trajectory representations and the relationship between trajectory representations in the latent space is performed in a deep learning neural network. In particular, a model based on the Transformer encoder and SOM is proposed. The Transformer encoder with the self-attention mechanism learns the internal characteristics of trajectories. To learn the relationship between the trajectory representations in the latent space, a SOM layer is used in our model. Specifically, the SOM learns clusters of trajectory representations in the latent space. In addition, in detecting anomalies, an effective mechanism is provided for determining an anomaly threshold. The anomaly threshold is first chosen based on trajectories' anomaly scores in the training set. Then, a new metric (i.e., WS) is proposed to select the appropriate value of the anomaly threshold.

III. PROBLEM DEFINITION

In indoor spaces, anomalous human movements can involve serious situations such as fire, violent attacks, and terrorism. Specifically, when a fire occurs suddenly in indoor spaces, humans tend to move following random routes to escape. In this case, their trajectories are anomalies compared with typical movement patterns. In addition, workers in a factory may be prohibited from accessing some locations like security control and engine rooms. Customers in supermarkets or stores are also not permitted to enter some places (i.e., security, staff areas, and warehouses). If one person visits these locations, his/her movement may be abnormal. Therefore, detecting anomalous human trajectories can improve safety in workplaces. This work aims at designing a framework for identifying abnormal human trajectories in indoor spaces.

Due to the requirement to detect such anomalies as soon as possible, trajectories are collected within a short time window, W, to check for abnormalities. A trajectory point p is the sampled position information at timestamp t, which is denoted (x, y, t) where (x, y) is the coordinates of p in the indoor space. For a given indoor space, every entity is assigned a semantic label (e.g., working room, meeting room, kitchen, corridor). Thus, each trajectory point is mapped to a semantic label, and we define point p as $((x, y), s, t)$, where s is the semantic label for p. Trajectory $T = \{p_1, p_2, \ldots p_n\}$ in which n is the point number of T in time window W.

A historical trajectory set $D = \{T_1, T_2, \ldots T_L\}$, which is people's location data in a specific indoor space, is given. The number of collected trajectories is L. Assume that T_{new} is a new trajectory, which comes during a time window W. We aim at detecting whether T_{new} is an anomaly based on the historical trajectory set D.

IV. METHODOLOGY

A. MODEL BASED ON THE TRANSFORMER ENCODER AND SOM

1) OVERVIEW OF THE PROPOSED MODEL

In this work, we build a model based on the Transformer encoder and SOM with two main tasks: learning normal trajectory representations and learning clusters of normal trajectory representations in latent space. The first task is based on the Transformer encoder, and the second is based on the SOM layer.

Since the proposed TENSO model captures interior features of normal trajectories, anomalous trajectories may not be well reconstructed. Moreover, anomalous trajectory representations in the latent space may not belong to clusters of normal representations, which are also learned by the TENSO model. Thus, the proposed model can detect anomalous trajectories using reconstruction errors and the relationship between representations and clusters of normal trajectory representations in latent space. The architecture of the TENSO model is divided into three main parts, as seen in Figure 2. The first part learns the input features of the trajectory (the green blocks). The second part includes the Transformer encoder and a decoder. The Transformer encoder learns trajectory representations and encodes them via output sequences in a latent space. A decoder with dense layers is used to reconstruct original trajectories. The third part is the SOM layer, which gets the input from the output of the Transformer encoder. This part learns normal trajectory representation clusters in latent space. Detailed descriptions of the parts are presented as follows.

2) LEARNING INPUT FEATURES OF TRAJECTORIES

To learn the correlation between a trajectory's points and to encode them using the Transformer encoder, the input

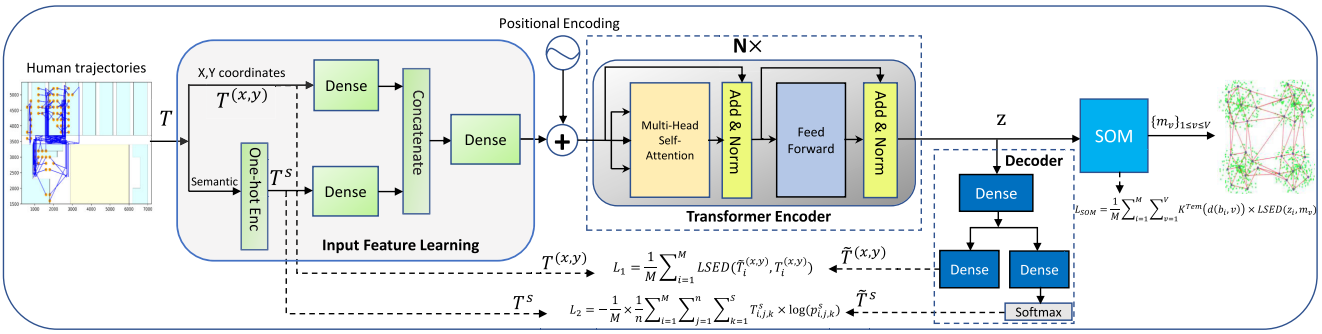

FIGURE 2. The architecture of the TENSO model.

trajectory points need to be embedded as vector representations. In particular, each trajectory point contains positional and semantic information. Since each trajectory's position belongs to a low-dimensional space (i.e., x and y coordinates), it is projected to a high-dimensional space. This helps the model learn more positional information about the trajectory points. This step is performed using a dense layer. Besides, since the x and y coordinate values may have different ranges, they are normalized to the range of [0,1] by using min-max normalization [35].

To use semantic information for learning trajectory representations, semantic labels need to be embedded as dense vectors. First, semantic labels of trajectory points are converted to one-hot vectors. Then, the one-hot vectors are projected to a high-dimensional space using a dense layer. Two vector representations of semantic and positional information are concatenated and projected in a higher dimension space to obtain the trajectory points' final vector representations.

3) LEARNING TRAJECTORY REPRESENTATIONS

In this subsection, the Transformer encoder and a decoder are presented. First, the Transformer encoder is used to learn the correlation between a trajectory's points and its sequential information. Then, the original trajectory reconstruction from the output sequence of the Transformer encoder is obtained using the decoder.

In the TENSO model, the Transformer encoder gets input as a sequence of trajectory points' embeddings. With the self-attention mechanism, each trajectory's point is encoded as a vector representation that captures the correlation of this point with all trajectory points. Since the attention mechanism does not retain the order of trajectory points by itself, positional encoding is added to the input of the Transformer encoder. The output of the Transformer encoder is a sequence of points' vector representations that captures internal characteristics and the sequential nature of the trajectory. The Transformer encoder consists of N stacked identical layers. Each layer comprises two main sub-layers: multi-head self-attention and position-wise fully connected feed-forward layer [27].

A simple decoder with dense layers is used to reconstruct the original trajectory, which ensures the trade-off between the trajectory reconstruction's effectiveness and the computational cost of the model. The decoder consists of one hidden layer and two outputs. The first output reconstructs the trajectory's positional information (i.e., x and y coordinates), and the second reconstructs the semantic label information (i.e., the probabilities that trajectory points' labels belong to semantic labels in the dataset).

4) SOM-BASED NORMAL TRAJECTORY REPRESENTATION CLUSTER MODELING

This subsection describes the SOM layer for learning normal trajectory representation clusters in latent space, which is trained jointly with the remaining parts of the TENSO model.

The correlations between trajectory representations in latent space may be useful for abnormal trajectory detection tasks. In particular, if a trajectory whose representation does not belong to any clusters in latent space, this trajectory may be an anomaly. Thus, the proposed model learns normal trajectory representation clusters in the latent space to detect anomalous trajectories, which can be performed by the SOM layer.

The input of the SOM is obtained from the output of the Transformer encoder. After training the SOM, the normal trajectory representation clusters in the latent space are modeled by prototype vectors of the SOM. Depending on the dataset and the application, the number of prototype vectors may vary. A small number of units on a map learns a general distribution of datasets, whereas a SOM with many units may represent more detailed datasets. In our work, the number of units is chosen as 100 (i.e., a square map of 10×10). This value is much smaller than the number of trajectories in the training sets (i.e., 10,537 in the MIT Badge dataset and 11,107 in the sCREEN dataset).

5) LOSS FUNCTIONS

In the TENSO model, all parameters are trained and updated jointly. A total loss function is designed that comprises three

different terms:

$$\text{Total Loss} = L_1 + \gamma_1 \times L_2 + \gamma_2 \times L_{\text{SOM}}, \tag{8}$$

where L_1 is the loss for reconstructing x-y coordinates, L_2 is the loss for reconstructing semantic labels, and L_{SOM} is the loss of the SOM; γ_1 and γ_2 are parameters that control the role of component loss functions in the total loss function.

In L_1, we use lock-step Euclidean distance (LSED) to determine the x-y coordinate reconstruction error between the decoded and original trajectories. L_1 is defined as

$$L_1 = \frac{1}{M}\sum_{i=1}^{M} \text{LSED}(\tilde{T}_i^{(x,y)}, T_i^{(x,y)}), \tag{9}$$

where M is the number of trajectories used for each parameter update, referred to as batch size. $\tilde{T}_i^{(x,y)}$ and $T_i^{(x,y)}$ are the decoded and original trajectories, respectively, which only contain x-y coordinate information of the trajectory points.

$\text{LSED}(\tilde{T}_i^{(x,y)}, T_i^{(x,y)})$ is defined as follows [36]:

$$\text{LSED}(\tilde{T}_i^{(x,y)}, T_i^{(x,y)}) = \frac{1}{n}\sum_{j=1}^{n} \text{dist}(\tilde{p}_j^{(x,y)}, p_j^{(x,y)}), \tag{10}$$

where $\text{dist}(\tilde{p}_j^{(x,y)}, p_j^{(x,y)})$ is the Euclidean distance between two points $\tilde{p}_j^{(x,y)}$ and $p_j^{(x,y)}$, which are the decoded and original trajectory points, and n is the point number of each trajectory.

In TENSO, since the model needs to predict the semantic label for each trajectory point, L_2 is chosen as the cross-entropy loss function, which is used for training models in multiclass classification. The equation for L_2 is

$$L_2 = -\frac{1}{M} \times \frac{1}{n}\sum_{i=1}^{M}\sum_{j=1}^{n}\sum_{k=1}^{S} T_{i,j,k}^{s} \times \log(p_{i,j,k}^{s}), \tag{11}$$

where S is the number of semantic label classes in the datasets, $T_{i,j,k}^{s}$ is the value of the semantic label information at the kth class of the jth point of the ith original trajectory (0 if a negative instance and 1 if a positive instance). $p_{i,j,k}^{s}$ is the probability that the semantic label of the jth point of the ith original trajectory belongs to the kth semantic label class.

For the loss of the SOM layer, L_{SOM} is defined by

$$L_{\text{SOM}} = \frac{1}{M}\sum_{i=1}^{M}\sum_{v=1}^{V} K^{\text{Tem}}(d(b_i, v)) \times \text{LSED}(z_i, m_v). \tag{12}$$

The terms in equation 12 are explained in Table 1.

The training procedure of the proposed model is presented in **Algorithm 1**.

TABLE 1. Explanation of terms.

Term	Meaning
M	The number of input trajectories
V	The number of units on the map
z_i	Latent representation of the ith input trajectory
b_i	Best matching unit of z_i
m_v	Prototype vector of unit v
$d(b_i, v)$	Manhattan distance between b_i and v
$\text{LSED}(z_i, m_v)$	Lock-step Euclidean distance between z_i and m_v
$K^{\text{Tem}}(.)$	The neighbor function

Algorithm 1 Training Procedure for TENSO

Input: - $X_{\text{Train}} = \{T_1, \ldots, T_{N_{\text{Train}}}\}$
- Number of epochs ($\text{num}_{\text{epoch}}$)
- Batch size (M)
- Temperatures $\text{Tem}_{\max}$, $\text{Tem}_{\min}$

Output: - Trained TENSO

1: Determine the number of iterations: $\text{Num}_{\text{iters}} = \text{num}_{\text{epoch}} \times [\frac{N_{\text{Train}}}{M}]$
2: **for** iter $\leftarrow 1$ **to** $\text{Num}_{\text{iters}}$ **do**
3: Get a batch of trajectories: $\{T_i\}_{i=1}^{M}$
4: Determine the latent representation batch: $\{z_i\}_{i=1}^{M}$
5: Determine the reconstructed terms of trajectories: $\{\tilde{T}_i^{(x,y)}\}_{i=1}^{M}$ and $\{\tilde{T}_i^{s}\}_{i=1}^{M}$
6: Find BMUs of the latent representation batch: $\{b_i\}_{i=1}^{M}$
7: Update temperature parameter Tem using equation 5
8: Determine neighborhood function $K^{\text{Tem}}(d(b_i, v))$ using equation 6
9: Update all parameters of TENSO using the total loss in equation 8
10: **end for**

B. ANOMALOUS TRAJECTORY DETECTION

This subsection discusses abnormal trajectory detection based on the TENSO model. In the training step of TENSO, normal historical trajectories are used. Then, a new trajectory is detected whether it is an anomaly. In particular, the anomaly score of each trajectory is determined based on the trained TENSO model. The trajectory is marked as an anomaly if the anomaly score exceeds a given threshold. Determination of the anomaly score and threshold are presented in detail as follows.

1) ANOMALY SCORE

To determine the anomaly score, the trajectory reconstruction errors and the quantization error of latent representation on the SOM are used. The trajectory's reconstruction errors consist of two terms: the x-y coordinate reconstruction error and the semantic label reconstruction error.

In two terms of the reconstruction errors, the x-y coordinate reconstruction error is measured using LSED distance, which is defined as the following equation.

$$\text{RE}_1 = \text{LSED}(\tilde{T}_i^{(x,y)}, T_i^{(x,y)}), \tag{13}$$

where $\tilde{T}_i^{(x,y)}$ is the x-y coordinate reconstruction of $T_i^{(x,y)}$.

To obtain the semantic label reconstruction error, the output of the softmax function from the decoder is processed. This output is a sequence of vectors. The jth vector contains probabilities that the semantic label of the jth point belongs to

label classes in dataset. The error label detection probability of the jth point in each trajectory may be determined as $(1 - p^s_{j,k=\text{Target label}})$. $p^s_{j,k=\text{Target label}}$ is correct label detection probability for the jth point. We define the trajectory's semantic label reconstruction error as the average value of trajectory point error detection probabilities, which is determined as follows:

$$\text{RE}_2 = \frac{1}{n}\sum_{j=1}^{n}(1 - p^s_{j,k=\text{Target label}}). \tag{14}$$

The quantization error is the distance between the latent representation of the trajectory and its BMU's prototype vector on the SOM [37]. In this work, the quantization error is also determined using LSED distance as follows:

$$\text{QE}_{\text{SOM}} = \text{LSED}(z_i, m_{b_i}), \tag{15}$$

where z_i is the latent representation of T_i, and m_{b_i} is the prototype vector of BMU b_i of z_i on the SOM.

The anomaly score of each trajectory is determined as

$$\text{AS} = (\alpha \times \text{RE}_1 + \beta \times \text{RE}_2) + (1 - \alpha - \beta) \times \text{QE}_{\text{SOM}}, \tag{16}$$

where α and β control the role of terms in the equation for AS. In this work, we set $\alpha = \beta = 0.25$. With these values, the role of the quantization error on the SOM and the reconstruction error by the Transformer encoder and decoder is the same (i.e., the weight is 0.5 for each term). Besides, the trajectory reconstruction errors for x-y coordinates and the semantic label are considered equally (i.e., the weight is 0.25 for each term).

Since the probability $p^s_{j,k=\text{Target label}}$ belongs to the range of [0,1], the value of RE_2 is also in the range of [0,1] in equation 14. However, the values of RE_1 and QE are determined using the LSED metric, and their ranges may differ from [0,1]. Thus, we normalize RE_1 and QE to the range of [0,1] using min-max normalization. This ensures all terms in the AS equation have the same range.

2) ANOMALY THRESHOLD

To detect abnormal trajectories, a threshold is used. In this work, a new method is proposed for determining the anomaly threshold based on anomaly scores of trajectories in the training set. The anomaly threshold is defined as follows:

$$\text{Thres} = \mu_{\text{AS}_\text{t}} + \theta \times \sigma_{\text{AS}_\text{t}}, \tag{17}$$

where μ_{AS_t} and $\sigma_{\text{AS}_\text{t}}$ are the mean and the standard deviation (SD) of the trajectory anomaly scores in the training set; θ is a parameter determined by the new WS metric. Note that since the training set only contains normal trajectories, Thres should be larger than μ_{AS_t}. Thus, θ should be positive.

The value of θ is chosen based on the proposed framework's performance on the validation set. The new WS factor, the weighted sum of recall and precision, is used for finding the appropriate value of θ. The equation for WS is defined as

$$\text{WS} = \delta \times \text{Recall} + (1 - \delta) \times \text{Precision}, \tag{18}$$

where δ is a parameter that controls the trade-off between recall and precision in WS. If δ is larger than 0.5, the role of recall is more important than precision; otherwise, precision is more important. In our work, δ is set at 0.5.

With an anomaly detection framework, it is expected that both recall and precision of the framework achieve high values. In this case, the WS of recall and precision also obtains a high value. This can only be achieved if the anomaly threshold is selected appropriately. If a chosen threshold is small, the framework tends to detect trajectories as anomalies. Thus, recall obtains a high value, whereas precision is small. In contrast, if the anomaly threshold is large, a trajectory is

Algorithm 2 Determine the Anomaly Threshold

Input: - Trained TENSO model
- $X_{\text{Train}} = \{T_{\text{train}_1}, \ldots, T_{\text{train}_{N_\text{T}}}\}$
- $X_{\text{Val}} = \{T_{\text{val}_1}, \ldots, T_{\text{val}_{N_\text{V}}}\}$
- Values $\theta_{\min}, \theta_{\max}$

Output: The chosen anomaly threshold: $\text{Thres}_{\text{chosen}}$

for $i \leftarrow 1$ **to** N_T **do**
  Determine $\text{RE}_1, \text{RE}_2, \text{QE}_{\text{SOM}}$ using TENSO with T_{train_i}
  Calculate $AS_{T_{\text{train}_i}}$ by equation 16
end for
for $i \leftarrow 1$ **to** N_V **do**
  Determine $\text{RE}_1, \text{RE}_2, \text{QE}_{\text{SOM}}$ using TENSO with T_{val_i}
  Calculate $AS_{T_{\text{val}_i}}$ by equation 16
end for
$\mu_{\text{AS}_\text{t}} \leftarrow \text{mean}\{\text{AS}_{T_{\text{train}_i}}\}_{i=\{1,\ldots,N_\text{T}\}}$
$\sigma_{\text{AS}_\text{t}} \leftarrow \text{SD}\{\text{AS}_{T_{\text{train}_i}}\}_{i=\{1,\ldots,N_\text{T}\}}$
for $\theta \leftarrow \theta_{\min}$ **to** $\theta_{\max}$ **do**
  $\text{Thres} \leftarrow \mu_{\text{AS}_\text{t}} + \theta \times \sigma_{\text{AS}_\text{t}}$
  for $i \leftarrow 1$ **to** N_V **do**
    if $\text{AS}_{T_{\text{val}_i}} \leq \text{Thres}$ **then**
      $T_{\text{val}_i} \leftarrow$ Normal trajectory
    else
      $T_{\text{val}_i} \leftarrow$ Abnormal trajectory
    end if
  end for
  Determine the number of True Positive (TP) samples in X_{Val}
  Determine the number of False Positive (FP) samples in X_{Val}
  Calculate recall and precision using TP, FP
  Calculate WS by equation 18
end for
$\theta_{\text{chosen}} \leftarrow \underset{\theta}{\text{argmax}}\ \text{WS}$
$\text{Thres}_{\text{chosen}} \leftarrow \mu_{\text{AS}_\text{t}} + \theta_{\text{chosen}} \times \sigma_{\text{AS}_\text{t}}$

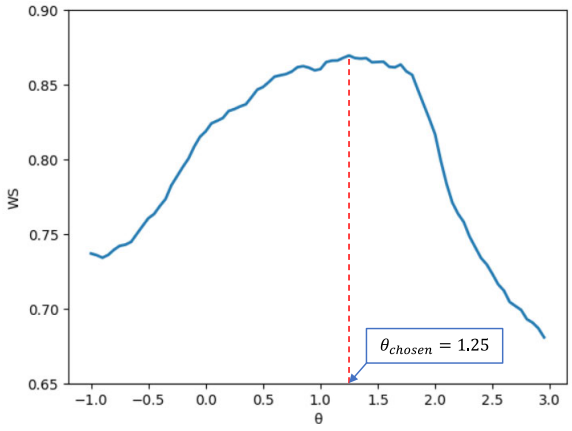

FIGURE 3. Choosing the value of θ.

marked as normal more easily. Therefore, recall is small, and precision is high. In both cases of threshold, the WS of recall and precision is small. From these aspects, an appropriate threshold is selected so that the framework achieves a maximum value for WS. In this work, θ is chosen according to the maximum value of WS on the validation set. It is expected that this value also yields a good performance with the test set. A detailed description for choosing θ and the anomaly threshold is in **Algorithm 2**. Figure 3 depicts the selected value for θ as 1.25, corresponding to the maximum WS value for the validation set in the MIT Badge dataset. The chosen value of θ is larger than 0, which is appropriate considering our previous statement about θ.

V. PERFORMANCE EVALUATION

A. EXPERIMENT SETUP

1) DATASETS

In this section, we use two real trajectory datasets to evaluate the proposed method: the MIT Badge dataset and the sCREEN dataset.

- **MIT Badge dataset.**
  The MIT Badge dataset is an indoor human location dataset [38]. It was collected from March 26 to April 17, 2007, at a Chicago-area data server configuration firm. Each recorded location contains information about x-y coordinates and timestamps. This dataset consists of three other groups with 39 workers. The largest one is the configuration group, with 28 workers. Besides, the pricing and the coordinator groups are smaller, with only seven and four workers, respectively. Since the configuration group has three workers with only a few days of data, those workers are dropped.
  In this work, a time window W of two minutes is used to collect data for each trajectory. Since the data sampling speed is 10 points per minute, the number of points in each trajectory is 20. Collecting data starts from 9:00 am to 6:00 pm daily. The number of collected days in the dataset is 17. The dataset is divided into three sets: nine days for training, three days for validation and five days for test. The days for each set are chosen randomly from the list of 17 days. A statistic about the number of trajectories is presented in Table 2. Note that the training and validation sets only consist of normal trajectories, which are used to train the model and choose the best model, respectively. However, to determine the anomaly threshold, the proposed method needs to be evaluated with the validation set. In this step, we need to have both normal and abnormal trajectories. In the validation set, the number of abnormal trajectories is 758, much smaller than the number of normal trajectories. To ensure the effectiveness of the performance evaluation, the number of normal and abnormal trajectories should be chosen equally. Thus, we randomly select 758 of 4,211 normal trajectories in the validation set for evaluating the performance to find the anomaly threshold. In the test set, the number of normal trajectories is also selected randomly to equal the number of abnormal trajectories (i.e., 1,614 for each trajectory type).

TABLE 2. Statistics of datasets.

Dataset		MIT Badge	sCREEN
Training set (Normal)		10,537	11,107
Validation set (Normal)		4,211	4,445
Test set	Normal	1,614	3,470
	Abnormal	1,614	3,470

- **sCREEN dataset.**
  The sCREEN dataset contains the customer location data collected from a German supermarket in July 2016 [16]. Like the MIT Badge dataset, each location point includes x-y coordinates and timestamps. We also gathered data for each trajectory using a time window W of two minutes. The data is available from 8:00 am to 10:00 pm each working day in the supermarket. This dataset was collected within 29 days with a large amount of 72,248 trajectories. Thus, we randomly selected five days from the list of collected days for the training set, two days for the validation set, and one day for the test set. A list of trajectories' numbers is also shown in Table 2.

2) CREATING ANOMALIES FOR EVALUATION

In the performance evaluation phase, anomalies are required. Since there are no anomalies in MIT Badge and sCREEN, they are generated and injected into the datasets for evaluation. In the literature, two methods are often used to make anomalies.

The first method is to hypothesize anomalies based on the different behavior groups in the dataset [39], [40]. In the MIT Badge dataset, the configuration group is the largest, with 28 workers, and the trajectories of workers in this group are assumed to be normal. In contrast, since the pricing

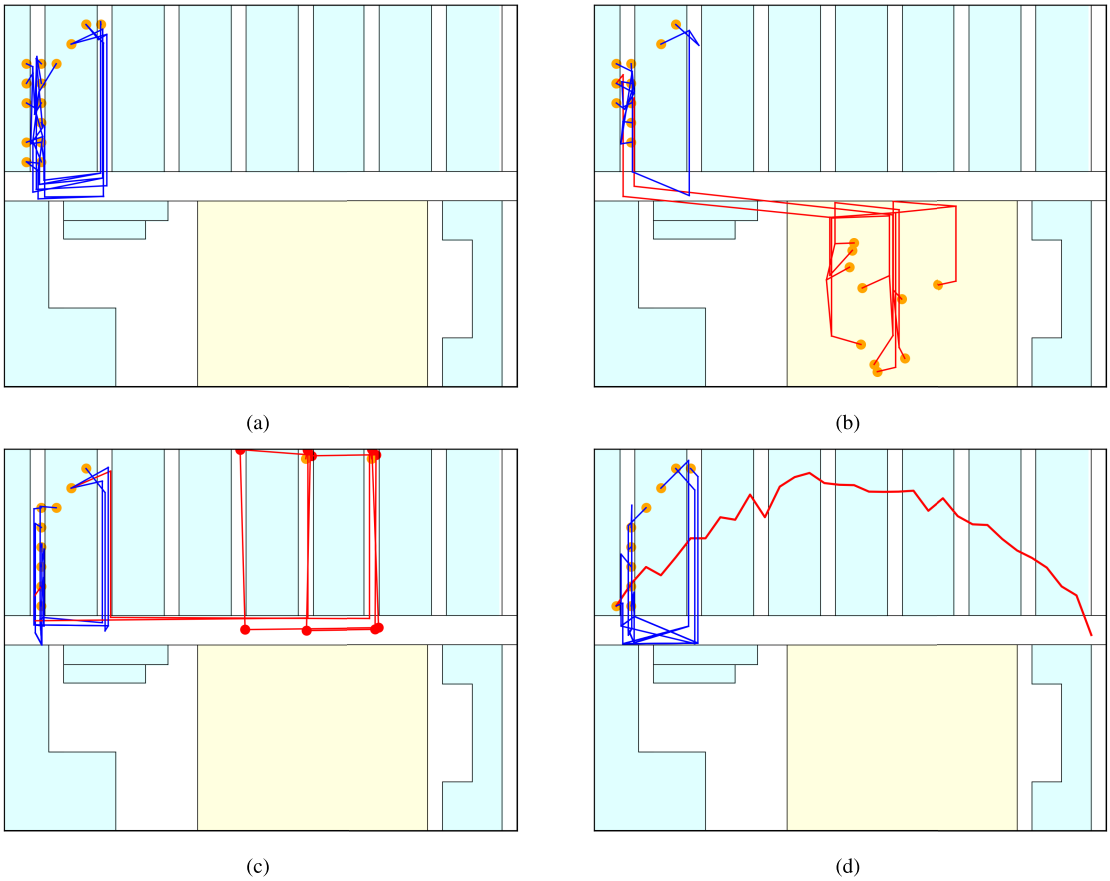

FIGURE 4. Creating anomaly types: (a) Normal original trajectory. (b) Rare location visiting anomaly. (c) Wandering anomaly. (d) Route anomaly.

group with only seven workers is much smaller than the configuration, trajectories were assumed to be abnormal. The sCREEN dataset does not include different behavior groups, so the first method is not applied to this dataset, and noise is injected into the validation set as abnormal samples to determine the anomaly threshold. The way to create noise was presented in our previous work [32]. The number of created noise samples equals the number of normal samples in the validation set (i.e., 4,445).

In the second method, anomalies are synthesized and injected into the datasets [41], [42], [43]. In our study, three anomaly types are generated for performance evaluation: Rare location visiting anomaly, wandering anomaly and route anomaly.

- **Rare location visiting anomaly.**
  Rare locations in an indoor space are where humans often do not enter or are prohibited from entering. People visit these locations only in urgent situations or for bad aims. Thus, if one person suddenly enters a rare location, their movements can be considered anomalies. To synthesize this anomaly type, the indoor space's rare locations first need to be determined. Rare locations in floor plans are marked if the historical trajectories did not travel to them. A location rare visiting anomaly can be created from a normal trajectory by shifting some sequential points to rare locations. To control the number of shifted points to the rare locations, parameter τ is used, and the start point for shifting the normal trajectory is selected randomly. Figure 4a and Figure 4b depict examples of a normal trajectory and a rare location visiting anomaly, respectively. In these figures, the yellow area represents a rare location. The blue path is the normal part, and the red path represents the anomaly part.
- **Wandering anomaly.** A wandering anomaly can be created when a person wanders around objects many times, possibly with bad intentions. Figure 4c is an example of a wandering anomaly around the working rooms (i.e., the red path). Parameter τ is also used to control the abnormality level.
- **Route anomaly.** A route anomaly occurs when people take unusual routes to escape in urgent situations (e.g., fire and attacking violence). In these cases, people tend to run to exits of the buildings. Thus, we choose the exit as the end point of the route anomaly. An example of

a route anomaly is generated as Figure 4d, with the red path being abnormal. Like the two above anomaly types, the abnormality level is controlled by parameter τ.

3) EVALUATION METRICS

In this work, we use three metrics: recall, precision and f1-score to estimate the algorithm performance. They are defined as follows:

$$\text{recall} = \frac{\text{True Positive}}{\text{Actual Anomalies}}, \tag{19}$$

$$\text{precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}, \tag{20}$$

$$\text{f1-score} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}, \tag{21}$$

where True Positive is the number of anomalies detected correctly. False Positive is the number of normal trajectories detected as anomalies, and Actual Anomalies is the number of anomalies in the dataset.

4) BASELINES

In this work, we compare the proposed method with five baselines.

- **Hierarchical clustering**. In this baseline, the hierarchical clustering algorithm was first applied to find the normal trajectory clusters in the dataset [31]. Then, a new trajectory can be detected as an anomaly based on its relationship with the found clusters.
- **DBSCAN**. The work in [32] detected anomalous trajectories using DBSCAN. In this work, a distance metric was proposed to measure the distance of trajectories, and the Epsilon parameter of DBSCAN was determined using a new DCVI metric. To find normal trajectory clusters in the dataset, DBSCAN was used. In detecting anomalous trajectories, if a trajectory did not belong to any clusters, it was marked as an anomaly.
- **LSTM-AE**. In [22], an autoencoder-based framework for detecting anomalies was proposed. The authors applied two autoencoders (CNN-AE and LSTM-AE) to learn the latent spaces from input sequences. We compare the proposed framework with the LSTM-AE-based framework in this baseline.
- **LSTM-VAE**. The authors in [44] proposed an anomaly detection framework with multi-sensor measurement values for maritime components. Their model contained an LSTM-based variational autoencoder (LSTM-VAE). The architecture of VAE constrained the latent space of input data to a standard normal distribution. Besides, they used the LSTM network in VAE to capture the temporal dependencies in the input sequences. In detecting anomalies, the reconstructed sequences from VAE were used to find the anomaly score. They determined an anomaly threshold based on the mean and standard deviation of anomaly scores for normal samples in the validation set.
- **Transformer**. The study in [34] introduced a Transformer-based anomaly detection framework for ECG signals. In this work, a deep learning model contains an embedding layer and a standard Transformer encoder to learn latent representations of original signals for detecting anomalies. Besides, an anomaly threshold was also determined based on predicted errors of normal signals in the training set.

B. RESULT ANALYSIS

The proposed model is implemented in Python 3.9.5 using the Tensorflow Keras library. To train the model, the Adam optimizer is used with a learning rate of 0.001. The number of training epochs is 40, and the batch size is 128. The model's hyper-parameters are selected based on the framework's performance on the validation set. Besides, parameters γ_1 and γ_2 in the total loss function are chosen based on the learning curves of components in this function. These experiments for selecting γ_1, γ_2 and the model hyper-parameters will be discussed in subsections V-B3 and V-B4, respectively. The set of selected parameters is listed in Table 3.

In the training phase, the best model is chosen at the epoch with the minimum validation loss. This means that choosing the model is based on the best result from learning trajectory representations and their clusters in latent space. With a given hyper-parameter set, selecting the best model in the training phase is independent of downstream tasks.

In the evaluation phase, the proposed framework uses both the MIT Badge and the sCREEN datasets. Besides, this work is compared with five baselines: hierarchical clustering-based, DBSCAN-based, LSTM-AE-based, LSTM-VAE-based, and Transformer-based methods. In the LSTM-AE-based method, since a mechanism for determining the anomaly threshold was not provided, we used the mechanism from the LSTM-VAE-based method to select an appropriate threshold.

Besides, two variants of the TENSO model are evaluated in this subsection. In particular, the first variant learns a latent space of normal trajectories using the Transformer encoder. Then, the SOM layer is used to learn normal trajectory representation clusters in the latent space. In this variant, the decoder is dropped, and trajectories are not reconstructed from the latent space. The variant is called TE-SOM. In the anomaly detection phase, the TE-SOM-based framework only uses the distance between the trajectory representation and its nearest cluster in the latent space.

In contrast, the second variant based on the Transformer encoder and a decoder (TE-Decoder) only learns normal trajectory representations. The SOM for learning normal trajectory representation groups is dropped in TE-Decoder. To detect anomalies, the TE-Decoder-based framework only uses trajectory reconstruction errors. The results of all methods for each anomaly type are presented below.

TABLE 3. List of parameters.

Parameters		Value	Studied in experiments
Input feature learning	Coordinate projecting dimension	64	✓
	Semantic projecting dimension	32	✓
Transformer encoder	d_{model}	128	✓
	Num of heads	4	✓
	Num of layers	4	✓
Decoder	Hidden dimension of decoder	64	✗
SOM	Map size of SOM	10x10	✓
	Tem_{max}	10	✗
	Tem_{min}	0.1	✗
Total loss function	γ_1	0.01	✓
	γ_2	0.001	✓
	Learning rate	0.001	✗
	Epochs	40	✗
	Batch size	128	✗

1) DETECTING PRICING GROUP AS ANOMALY

Since the occurrence frequency and movement behaviors in worker groups are different in the MIT Badge dataset, a hypothesis is given to assign abnormal and normal labels to trajectories in this dataset [39]. Specifically, if the occurrence frequency of a group is much less than other groups, it can be considered an anomaly group. In the MIT Badge dataset, the pricing group accounts for only about 18% of the total, and the configuration group is approximately 72%. Therefore, workers' trajectories in the pricing and configuration groups are assigned as abnormal and normal samples, respectively. Table 4 shows the results of all methods when detecting pricing group as anomaly in the MIT Badge dataset. We can see that TENSO achieves 89.64% in terms of f1-score and outperforms all baselines.

In particular, the TENSO-based framework is significantly better than clustering-based methods (i.e., about 19% better than hierarchical clustering and 6% better than DBSCAN). With the clustering-based baselines, anomaly detection was only based on the relationship between the test sample and clusters in the dataset. In other words, they did not discover the internal characteristics of trajectories. In contrast, TENSO jointly learns the correlation between points within each trajectory and trajectory representation clusters in the latent space to detect anomalies. Therefore, the proposed framework discovers more helpful information about trajectories and improves accuracy in detecting anomalies. Note that the DBSCAN-based method in [32] used a time window of 10 minutes to collect data for each trajectory. In this work, to detect anomalies earlier, a shorter time window of only 2 minutes is used. Since the trajectory length is smaller, determining whether the trajectory is normal or abnormal becomes more difficult and leads to lower performance.

Besides, our method detects anomalies more effectively than the existing deep learning-based methods. In particular, TENSO outperforms LSTM-AE, LSTM-VAE and Transformer at 10.55%, 12.31% and 9.83%, respectively, in terms of f1-score. These baselines focused on learning a latent space that captures the internal characteristics of trajectories. Anomalies were detected using reconstructed trajectories through the models. However, the three baselines did not discover the clusters of trajectory representations in the latent space for anomaly detection, which is performed in the TENSO model. This explains that the TENSO-based framework achieves better performance than the baselines.

TABLE 4. Results for detecting pricing group as anomaly in MIT Badge dataset.

Method	Recall	Precision	f1-score
Hierarchical clustering	0.7368	0.6708	0.7009
DBSCAN	0.8563	0.8149	0.835
LSTM-AE	0.8798	0.7183	0.7909
LSTM-VAE	0.7695	0.7772	0.7733
Transformer	0.7924	0.8039	0.7981
TE-SOM	0.9275	0.7595	0.8351
TE-Decoder	0.8066	0.8744	0.8392
TENSO	0.9009	0.892	**0.8964**

The TE-SOM and TE-Decoder variants achieve similar results in terms of f1-score (i.e., about 84%). In other aspects, TE-SOM obtains the highest recall at 92.75 % while only achieving approximately 76% for precision. In contrast, TE-Decoder outperforms TE-SOM in precision by about 11%. Thus, combining the two variants in TENSO helps achieve the best anomaly detection performance where both recall and precision are high.

2) DETECTING SYNTHESIZED ANOMALIES

Three anomaly types (i.e., rare location visiting anomaly, wandering anomaly and route anomaly) are injected into

TABLE 5. Results for detecting synthesized anomalies in the MIT Badge dataset.

Synthesized anomalies	τ	0.5			0.75			1		
	Method	Recall	Precision	f1-score	Recall	Precision	f1-score	Recall	Precision	f1-score
Rare location visiting anomaly	Hierarchical clustering	0.9715	0.6892	0.8064	1	0.6954	0.8203	1	0.6954	0.8203
	DBSCAN	0.6579	0.7718	0.7104	1	0.8371	0.9113	1	0.8371	0.9113
	LSTM-AE	0.9988	0.7432	0.8522	1	0.7434	0.8528	1	0.7434	0.8528
	LSTM-VAE	0.9628	0.8136	0.882	0.9895	0.8177	0.8954	0.9994	0.8192	0.9004
	Transformer	0.9988	0.8378	0.9112	0.9994	0.8379	0.9116	1	0.838	0.9119
	TE-SOM	1	0.7729	0.872	1	0.7729	0.872	1	0.7729	0.872
	TE-Decoder	1	0.8961	0.9452	1	0.8961	0.9452	1	0.8961	0.9452
	TENSO	1	0.9017	**0.9483**	1	0.9017	**0.9483**	1	0.9017	**0.9483**
Route anomaly	Hierarchical clustering	0.5713	0.566	0.5686	0.938	0.6817	0.7896	0.9418	0.6825	0.7915
	DBSCAN	0.544	0.7366	0.6258	0.9789	0.8342	0.9008	0.9957	0.8365	0.9092
	LSTM-AE	0.8854	0.7195	0.7939	0.9882	0.7412	0.8471	0.995	0.7425	0.8504
	LSTM-VAE	0.8841	0.8003	0.8402	0.9281	0.808	0.8639	0.9622	0.8135	0.8816
	Transformer	0.8742	0.8189	0.8456	0.9368	0.8289	0.8796	0.9851	0.836	0.9044
	TE-SOM	0.9039	0.7547	0.8227	0.964	0.7665	0.854	0.9962	0.7723	0.8701
	TE-Decoder	0.8296	0.8774	0.8529	0.9368	0.8899	0.9128	0.9857	0.8948	0.9381
	TENSO	0.855	0.8869	**0.8707**	0.9932	0.9011	**0.9449**	0.9975	0.9014	**0.9471**
Wandering anomaly	Hierarchical clustering	0.6406	0.5939	0.6164	0.7429	0.6291	0.6813	0.7745	0.6387	0.7001
	DBSCAN	0.7534	0.7948	0.7735	0.9994	0.8371	0.911	1	0.8371	0.9113
	LSTM-AE	0.9994	0.7433	0.8525	1	0.7434	0.8528	1	0.7434	0.8528
	LSTM-VAE	0.8618	0.7962	0.8277	0.9449	0.8107	0.8727	0.995	0.8186	0.8982
	Transformer	0.9473	0.8305	0.8851	0.9752	0.8346	0.8994	0.9777	0.8349	0.9007
	TE-SOM	0.9993	0.7728	0.8717	1	0.7729	0.872	1	0.7729	0.872
	TE-Decoder	1	0.8961	0.9452	1	0.8961	0.9452	1	0.8961	0.9452
	TENSO	1	0.9017	**0.9483**	1	0.9017	**0.9483**	1	0.9017	**0.9483**

both datasets for evaluation. We also change the τ parameter to control the abnormality level of synthesized trajectories. In particular, τ is set to 0.5, 0.75 and 1 according to the time for anomaly part in each trajectory being 1, 1.5 and 2 minutes, respectively.

Tables 5 and 6 present the results of all methods when detecting three synthesized anomaly types in the MIT Badge and sCREEN datasets, respectively. The proposed framework achieves the best performance in these experiments compared with all the baselines. In particular, with the MIT Badget dataset, the TENSO-based framework detects all rare location visiting and wandering anomalies for three τ settings and achieves an f1-score of 94.83%. With route anomalies, the f1-score is achieved at 87.07%, 94.49% and 94.71% according to τ be 0.5, 0.75 and 1, respectively. With the sCREEN dataset, the proposed framework also achieves results similar to the MIT Badge dataset. All rare location visiting anomalies are identified, and the f1-score is approximately 96% in this case. The results for detecting route and wandering anomalies are also high, with f1-score at about 91% and 93%, respectively, for τ of 0.5. When τ increases to 0.75 and 1, f1-score is about 95% for both anomaly types. In both datasets, it can be seen that when τ increases, the anomaly detection accuracy of all methods is also larger for all anomaly types. Since abnormality in trajectories is higher when τ increases, they are detected more easily. Thus, the anomaly detection results are higher.

In the clustering-based baselines, when $\tau = 0.5$, the anomaly detection performance is much lower than the deep learning-based methods in both datasets. For example, at $\tau = 0.5$, the TENSO-based framework outperforms the hierarchical clustering-based method by about 30% for route and wandering anomalies in the MIT Badge dataset and for all anomaly types in the sCREEN dataset. This can be explained that the clustering-based detection methods' performance was affected by the distance metric of trajectories. In these baselines, the distance metrics (i.e., the longest common subsequence (LCSS) in the hierarchical clustering-based method and the extension of LCSS (LCSS_IS) in the DBSCAN-based method) were used. LCSS and LCSS_IS aim at finding the largest number of similar points between two trajectories, ignoring unmatched points. In other words, these metrics tend to find normality rather than abnormality [32]. Thus, if the anomaly part in synthesized trajectories is small, the anomaly detection performance of clustering-based methods is low.

TABLE 6. Results for detecting synthesized anomalies in the sCREEN dataset.

Synthesized anomalies	τ	0.5			0.75			1		
	Method	Recall	Precision	f1-score	Recall	Precision	f1-score	Recall	Precision	f1-score
Rare location visiting anomaly	Hierarchical clustering	0.9677	0.5176	0.6745	1	0.5258	0.6892	1	0.5258	0.6892
	DBSCAN	0.7948	0.739	0.7659	1	0.7808	0.8769	1	0.7808	0.8769
	LSTM-AE	1	0.7942	0.8853	1	0.7942	0.8853	1	0.7942	0.8853
	LSTM-VAE	1	0.8019	0.8901	1	0.8019	0.8901	1	0.8019	0.8901
	Transformer	1	0.871	0.931	1	0.871	0.931	1	0.871	0.931
	TE-SOM	0.9625	0.819	0.885	0.9916	0.8234	0.8997	0.998	0.8243	0.9029
	TE-Decoder	1	0.9093	0.9525	1	0.9093	0.9525	1	0.9093	0.9525
	TENSO	1	0.9214	**0.9591**	1	0.9214	**0.9591**	1	0.9214	**0.9591**
Route anomaly	Hierarchical clustering	0.9375	0.5096	0.6603	0.9977	0.5252	0.6882	1	0.5258	0.6869
	DBSCAN	0.6772	0.707	0.6918	1	0.7808	0.8769	1	0.7808	0.8769
	LSTM-AE	0.9008	0.7766	0.8342	0.9011	0.7767	0.8343	0.9034	0.7771	0.8356
	LSTM-VAE	0.7654	0.756	0.7654	1	0.8019	0.8901	1	0.8019	0.8901
	Transformer	0.936	0.8634	0.8982	0.9997	0.871	0.9309	1	0.871	0.931
	TE-SOM	0.9974	0.8242	0.9026	0.9974	0.8242	0.9026	0.9974	0.8242	0.9026
	TE-Decoder	0.8406	0.894	0.8665	0.9594	0.9059	0.9318	0.9948	0.9089	0.9499
	TENSO	0.9201	0.9151	**0.9177**	0.989	0.9206	**0.9536**	0.9925	0.9208	**0.9553**
Wandering anomaly	Hierarchical clustering	0.8671	0.4901	0.6262	0.9392	0.5101	0.6611	0.9524	0.5136	0.6673
	DBSCAN	0.8075	0.7421	0.7734	1	0.7808	0.8769	1	0.7808	0.8769
	LSTM-AE	0.9302	0.7821	0.8498	1	0.7942	0.8853	1	0.7942	0.8853
	LSTM-VAE	0.9484	0.7934	0.864	1	0.8019	0.8901	1	0.8019	0.8901
	Transformer	0.9945	0.8704	0.9283	0.998	0.8708	0.93	1	0.871	0.931
	TE-SOM	0.9107	0.8107	0.8578	0.9326	0.8143	0.8694	0.9974	0.8242	0.9026
	TE-Decoder	0.9971	0.9091	**0.9511**	1	0.9093	**0.9525**	1	0.9093	0.9525
	TENSO	0.944	0.9171	0.9304	0.9674	0.9189	0.9426	0.9948	0.921	**0.9565**

Moreover, as stated previously, the clustering-based baselines only use the relationship between the test trajectory and the clusters of normal trajectories in the dataset to detect anomalies. They did not explore the internal characteristics of the trajectories. Therefore, their performance is still lower than the proposed framework even when the abnormality in synthesized trajectories increases (e.g., $\tau = \{0.75, 1\}$).

In the existing deep learning-based methods, the best model (i.e., Transformer) achieves 91.19% and 93.1% in terms of f1-score for detecting rare location visit anomalies in the MIT Badge and sCREEN datasets, respectively. The results are still lower than the TENSO-based framework by 3.64% with the MIT Badge dataset and by 2.55% with the sCREEN dataset. The reason for the performance difference is that the baselines were solely based on trajectory reconstruction errors to detect anomalies. In contrast, the proposed framework uses both trajectory reconstruction error and quantization error on SOM. Tables 5 and 6 also show that the Transformer-based method is better than LSTM-AE-based and LSTM-VAE-based methods on both datasets in terms of f1-score. This can be explained that the Transformer architecture is more effective than the LSTM architecture in learning trajectory representations for detecting anomalies.

With the two variants of the proposed model, TE-Decoder achieves high precision while TE-SOM maintains high recall for all anomaly types. Therefore, the combination of TE-Decoder and TE-SOM in TENSO significantly improves detection results with both datasets, as shown in Tables 5 and 6. Besides, as can be seen, the TE-Decoder also has an attractive performance in f1-score for all anomalies. This variant is even better than the TENSO-based method when detecting the wandering anomaly in the sCREEN dataset: about 2% and 1% higher for f1-score when $\tau = 0.5$ and 0.75, respectively. This means that learning internal characteristics and sequential information of trajectories based on the Transformer encoder and decoder plays an important role in detecting anomalies. In addition, as can be seen from Tables 5 and 6, the TE-Decoder variant also outperforms Transformer-based method, although both models also use the Transformer architecture. This is because an effective way to select the anomaly threshold is provided in the TE-Decoder. In contrast, in the Transformer-based method, a fixed value is used in the formulation of determining the anomaly threshold, which affects the performance of detecting anomalies.

From Tables 5 and 6, it can be shown that the deep learning-based methods can easily detect the rare location visiting anomaly. This is possibly explained that this anomaly

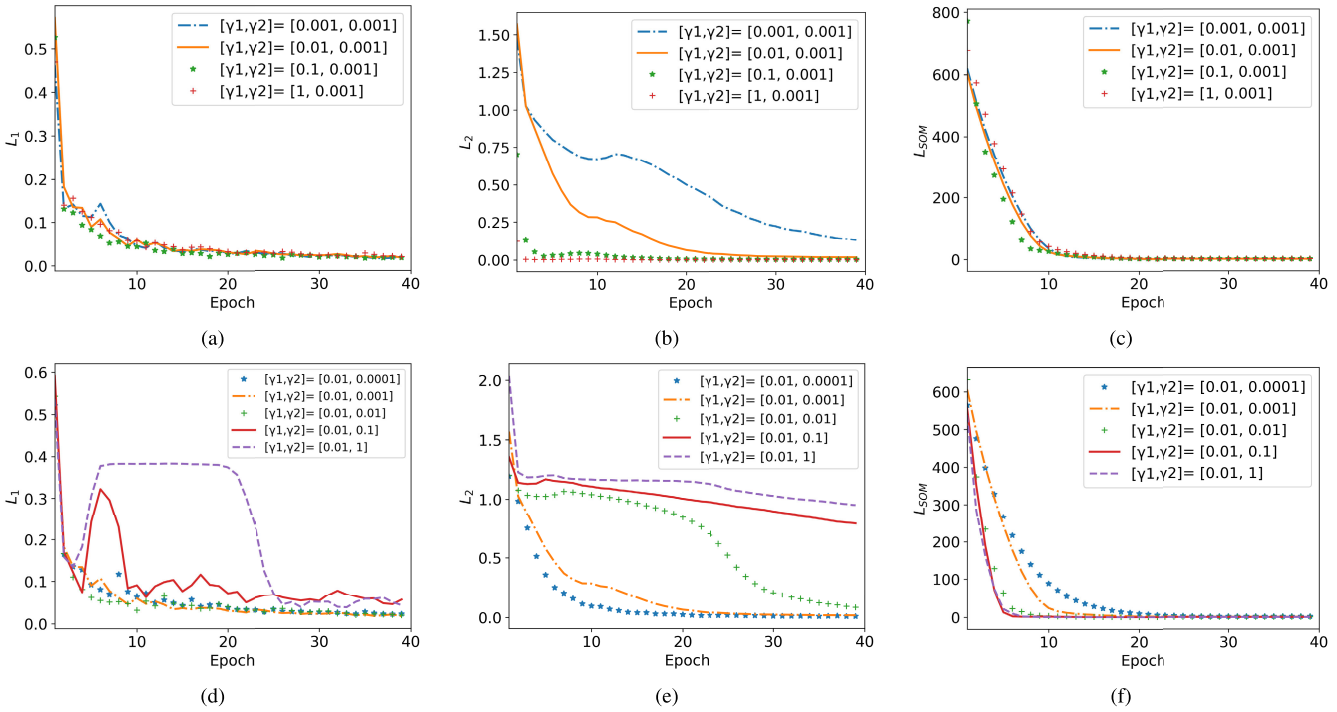

FIGURE 5. Component loss functions when changing γ_1 and γ_2: (a) L_1 when changing γ_1. (b) L_2 when changing γ_1. (c) L_{SOM} when changing γ_1. (d) L_1 when changing γ_2. (e) L_2 when changing γ_2. (f) L_{SOM} when changing γ_2.

type contains rare locations, which the deep learning models do not see in the training phase. Thus, the rare location visiting anomaly is not reconstructed well by the models, and they are detected more easily than the two remaining anomaly types. For example, the lowest recall still achieves 96.28% for LSTM-VAE in the MIT Badge dataset and 96.25% for TE-SOM in the sCREEN dataset. With the TENSO-based framework, all anomalies of this type are detected.

Note that if the anomaly detection ability of a method does not change over different anomaly types, the f1-score of the method is the same for the anomaly types. Since the anomaly detection ability is maintained, the number of true positive samples does not change over the anomaly types. Besides, since normal samples are maintained when evaluating the methods for all anomaly types, the number of false positive samples is also kept. Thus, the f1-score, determined using the number of true positive and false positive samples, is the same for the anomaly types. For example, in the MIT Badge dataset, since all samples of rare location visiting and wandering anomalies are correctly detected by the TENSO-based framework, the results in terms of the f1-score for this method are the same for the two anomaly types.

3) SENSITIVITY OF PARAMETERS IN LOSS FUNCTIONS

This subsection discusses the γ_1 and γ_2 parameters that control the trade-off between components in the total loss function of the proposed model. The experiments are performed with the MIT Badge dataset.

The influence of these parameters on L_1, L_2, and L_{SOM} (i.e., the loss function for reconstructing x-y coordinates, the loss function for reconstructing semantic information, and the loss of the SOM, respectively) is shown in Figure 5. As can be seen in Figure 5, L_1 has the smallest amplitude. The amplitude of L_{SOM} is the largest and is much larger than the two remaining components. Note that the weight of L_1 in the total loss function is 1. Thus, to ensure the trade-off between components in the total loss function, the weight of L_2 (i.e., γ_1) should be smaller than 1, and the weight of L_{SOM} (i.e., γ_2) should be smaller than γ_1.

To evaluate the effect of γ_1 on the learning curves, γ_2 is kept to a value smaller than 1 (e.g., 0.001), and we change γ_1 to different values $\{0.001, 0.01, 0.1, 1\}$. The top row of Figure 5 shows the evolution of three learning curves in the total loss function when changing γ_1. In Figure 5a, the learning curves of L_1 according to values of γ_1 are only slightly different. This can be explained that since the amplitude of L_2 is not much larger than L_1, the effect of changing the weight of L_2 on the learning curve of L_1 is unclear. Besides, as can be seen in Figure 5c, optimization of the SOM loss is ensured with each value of γ_1. As previously shown, the amplitude of L_2 is much smaller than L_{SOM}. Thus, the optimizing process of L_{SOM} is also not influenced by γ_1. In contrast, since γ_1 is the weight of L_2, it directly affects the learning curve of L_2. In particular, the optimizing process is slow when the weight is too small (e.g., when $\gamma_1 = 0.001$). When the weight is near or equal to 1, the learning curve is not smooth and quickly drops to the optimal value in the first few training epochs. Therefore,

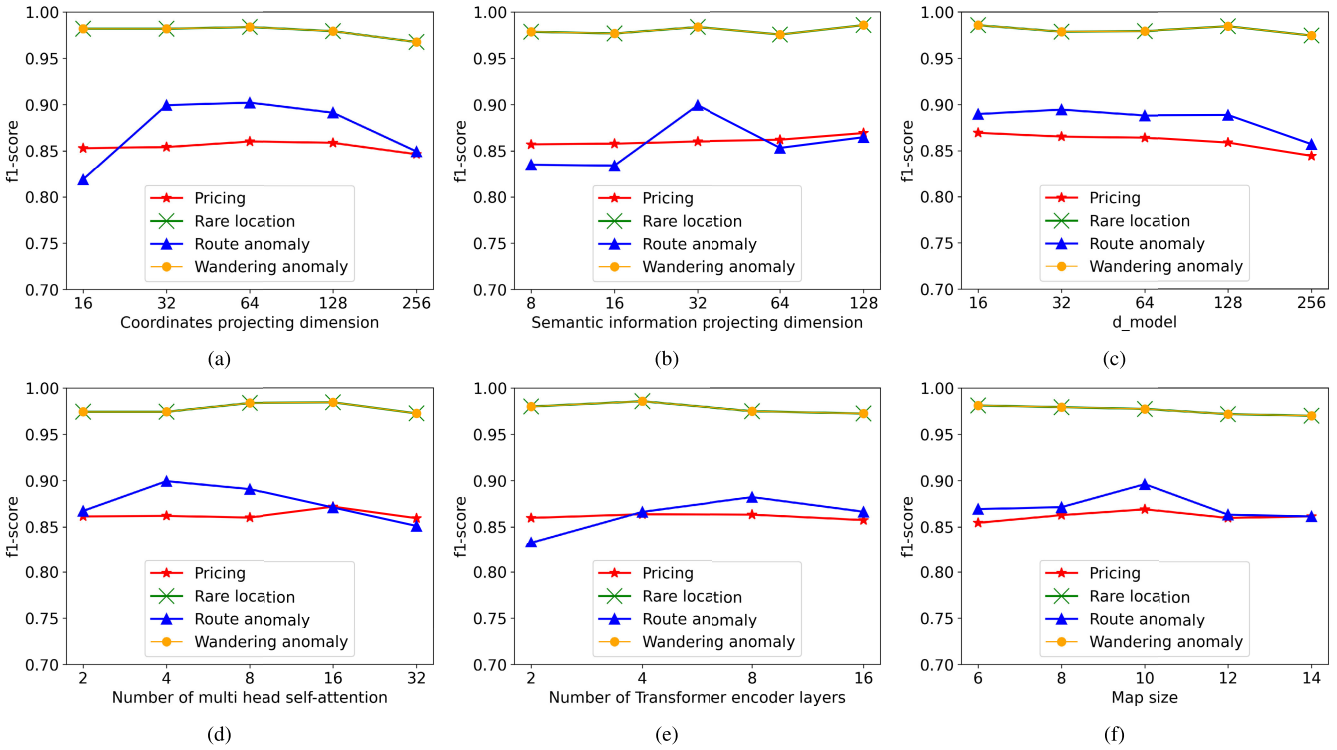

FIGURE 6. Influence of parameters on the performance in MIT Badge dataset: (a) Coordinates projecting dimension. (b) Semantic projecting dimension. (c) D-model. (d) Number of multi head self-attention. (e) Number of Transformer encoder layers. (f) Map size.

to ensure the optimization of all components in the total loss function, an appropriate γ_1 value of 0.01 is chosen.

Next, the evolution of learning curves when changing γ_2 is shown in the bottom row of Figure 5. Note that in the experiments, $\gamma_1 = 0.01$. As can be seen, the L_1 and L_2 reconstruction losses are clearly affected when γ_2 is changed. That is possibly explained that the value of L_{SOM} is much higher than L_1 and L_2. Thus, when the weight of L_{SOM} changes, it influences these losses remarkably. In particular, with a value for γ_2 of 0.1 or 1, the optimizing process of L_1 is improper, as seen in Figure 5d. It decreases quickly after the first few epochs and then increases significantly in the subsequent epochs before reaching the optimal value. When γ_2 is reduced to one of the $\{0.0001, 0.001, 0.01\}$ values, the optimization of this function is ensured. Similarly, the L_2 reconstruction loss function is not optimized correctly with γ_2 from the set $\{0.01, 0.1, 1\}$ as seen in Figure 5e. With the SOM loss, γ_2 has more influence than γ_1 on the learning curve. The reason is that γ_2 is the weight of L_{SOM}, directly affecting this learning curve. However, optimization of L_{SOM} is still properly ensured with all the values of γ_2. From the above analysis, we select 0.001 as the γ_2 value, which is appropriate for the three loss functions.

4) EFFECTS OF HYPER-PARAMETERS

A discussion about the effects of model hyper-parameters on the proposed framework's performance is given in this subsection. Specifically, Figure 6 shows the effects of six hyper-parameters: coordinate projecting dimension, semantic projecting dimension, d_{model} (i.e., the input dimension of the Transformer encoder), the number of heads in the multi-head self-attention, the number of layers in the Transformer encoder, and the map size of the SOM layer. The experiments are performed on the validation set of the MIT Badge dataset, and both hypothesized and synthesized anomalies are used for evaluation. Note that the synthesized anomalies are generated with $\tau = 1$ in the experiments. The proposed framework's performance is presented in terms of the f1-score.

First, as can be seen in Figures 6a and 6b, the route anomaly detection ability by the framework is clearly affected when changing semantic information and coordinates projecting dimensions of trajectory points. In contrast, with the remaining anomalies, the framework performance is stable over the different values for these hyper-parameters. Thus, selecting semantic information and coordinate projecting dimensions is based on the performance from detecting the route anomaly. In particular, the semantic information and the coordinate projecting dimensions are set to 32 and 64, respectively. The proposed framework achieves the best performance on the route anomaly at these values.

Next, Figures 6c, 6d and 6e present the effects on the framework's performance from hyper-parameters on the Transformer encoder (i.e., the d_model, the number for multi-head self-attention, and the number of Transformer encoder layers, respectively). In Figure 6c, framework performance

is stable over all anomaly types when the d_model is not greater than 128. If $d_{\text{model}} = 256$, the performance decreases. One possible reason is that when the d_{model} is too large, the model becomes more complex and causes the over-fitting problem. In the case, the value of the d_{model} is set at 128. From Figures 6d and 6e, the number of heads in the multi-head self-attention and the number of Transformer encoder layers are set at 4. These values are chosen to ensure the trade-off between effectiveness and computational cost.

Finally, the effect of map size on the SOM is shown in Figure 6f. As can be seen in Figure 6f, the framework's performance when changing the map size is also stable for hypothesized anomalies (i.e., pricing group's trajectories) and the two synthesized anomaly types (i.e., rare location and wandering anomalies). Therefore, choosing the map size is also based on the proposed framework's performance when detecting route anomalies. From Figure 6f, the map size is chosen at 10 according to the best performance on the route anomaly.

VI. CONCLUDING REMARKS

This work proposed an anomalous indoor human trajectory detection framework using a deep learning model. Our proposed model was based on the Transformer encoder and SOM, which jointly learned normal trajectory representations and their clusters in the latent space. In particular, the Transformer encoder with a self-attention mechanism was used to learn the correlations between points within each trajectory and its sequence information. A decoder reconstructed the input trajectory from its latent representation. In addition, the SOM layer was used to learn clusters of normal trajectory representations in the latent space. In detecting anomalous trajectories, the anomaly score of the test trajectory was determined by using the trained model. The trajectory's anomaly score contained the trajectory reconstruction errors from the latent space and the quantization error on the SOM. If the anomaly score exceeded a set threshold, the test trajectory was detected as an anomaly. In this work, we also proposed a novel metric called WS, the weighted sum of recall and precision for determining the anomaly threshold. Especially an appropriate anomaly threshold was selected according to the maximum value for WS with the validation set. Our framework was estimated using two real trajectory datasets: MIT Badge and sCREEN. The results depicted that the proposed framework achieved attractive performance and outperformed existing methods in anomaly trajectory detection.

In the future, we will extend the proposed model to learn more information about trajectories (e.g., speed, direction) to improve anomaly detection performance. Besides, the model will be developed to detect the anomaly trajectories even if the trajectories are not complete.

REFERENCES

[1] P. Kothari and A. Alahi, "Safety-compliant generative adversarial networks for human trajectory forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4251–4261, Apr. 2023.

[2] S. Saadatnejad, M. Bahari, P. Khorsandi, M. Saneian, S.-M. Moosavi-Dezfooli, and A. Alahi, "Are socially-aware trajectory prediction models really socially-aware?" *Transp. Res. C, Emerg. Technol.*, vol. 141, Aug. 2022, Art. no. 103705.

[3] Q. T. Ngo, D. T. Lan, S. Yoon, W.-S. Jung, T. Yoon, and D. Yoo, "Companion mobility to assist in future human location prediction," *IEEE Access*, vol. 10, pp. 68111–68125, 2022.

[4] J. Wang, N. Wu, W. X. Zhao, F. Peng, and X. Lin, "Empowering A search algorithms with neural networks for personalized route recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Anchorage, AK, USA, Jul. 2019, pp. 539–547.

[5] G. Cui, J. Luo, and X. Wang, "Personalized travel route recommendation using collaborative filtering based on GPS trajectories," *Int. J. Digit. Earth*, vol. 11, no. 3, pp. 284–307, Mar. 2018.

[6] J. Dai, B. Yang, C. Guo, and Z. Ding, "Personalized route recommendation using big trajectory data," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Seoul, South Korea, Apr. 2015, pp. 543–554.

[7] E. Kanoulas, Y. Du, T. Xia, and D. Zhang, "Finding fastest paths on a road network with speed patterns," in *Proc. 22nd Int. Conf. Data Eng. (ICDE)*, Atlanta, GA, USA, 2006, p. 10.

[8] C.-C. Hung, W.-C. Peng, and W.-C. Lee, "Clustering and aggregating clues of trajectories for mining trajectory patterns and routes," *VLDB J.*, vol. 24, no. 2, pp. 169–192, Apr. 2015.

[9] M. Nanni and D. Pedreschi, "Time-focused clustering of trajectories of moving objects," *J. Intell. Inf. Syst.*, vol. 27, no. 3, pp. 267–289, Nov. 2006.

[10] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: A partition-and-group framework," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Beijing, China, Jun. 2007, pp. 593–604.

[11] J. Lan, C. Long, R. C.-W. Wong, Y. Chen, Y. Fu, D. Guo, S. Liu, Y. Ge, Y. Zhou, and J. Li, "A new framework for traffic anomaly detection," in *Proc. SIAM Int. Conf. Data Mining*, Philadelphia, PA, USA, Apr. 2014, pp. 875–883.

[12] A. Belhadi, Y. Djenouri, G. Srivastava, D. Djenouri, A. Cano, and J. C. Lin, "A two-phase anomaly detection model for secure intelligent transportation ride-hailing trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4496–4506, Jul. 2021.

[13] Y. Wang, K. Qin, Y. Chen, and P. Zhao, "Detecting anomalous trajectories and behavior patterns using hierarchical clustering from taxi GPS data," *ISPRS Int. J. Geo-Inf.*, vol. 7, no. 1, p. 25, Jan. 2018.

[14] S. Calderara, U. Heinemann, A. Prati, R. Cucchiara, and N. Tishby, "Detecting anomalies in people's trajectories using spectral graph analysis," *Comput. Vis. Image Understand.*, vol. 115, no. 8, pp. 1099–1111, Aug. 2011.

[15] S. Wu, B. E. Moore, and M. Shah, "Chaotic invariants of Lagrangian particle trajectories for anomaly detection in crowded scenes," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, San Francisco, CA, USA, Jun. 2010, pp. 2054–2060.

[16] M. Paolanti, D. Liciotti, R. Pietrini, A. Mancini, and E. Frontoni, "Modelling and forecasting customer navigation in intelligent retail environments," *J. Intell. Robot. Syst.*, vol. 91, no. 2, pp. 165–180, Aug. 2018.

[17] P. Wang, J. Yang, and J. Zhang, "Location prediction for indoor spaces based on trajectory similarity," in *Proc. 4th Int. Conf. Data Sci. Inf. Technol.*, Shanghai, China, Jul. 2021, pp. 402–407.

[18] P. Wang, J. Yang, and J. Zhang, "A spatial–temporal-semantic method for location prediction in indoor spaces," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–13, Mar. 2022.

[19] L. Bao, S. Wu, W. Chen, Z. Zhu, and F. Yi, "Trajectory outlier detection based on partition-and-detection framework," in *Proc. 13th Int. Conf. Natural Comput., Fuzzy Syst. Knowl. Discovery (ICNC-FSKD)*, Cancun, Mexico, Jul. 2017, pp. 1978–1983.

[20] J. Zhu, W. Jiang, A. Liu, G. Liu, and L. Zhao, "Time-dependent popular routes based trajectory outlier detection," in *Proc. 16th Int. Conf. Web Inf. Syst. Eng.*, Miami, FL, USA, Nov. 2015, pp. 16–30.

[21] L. Song, R. Wang, D. Xiao, X. Han, Y. Cai, and C. Shi, "Anomalous trajectory detection using recurrent neural network," in *Advanced Data Mining and Applications*. Nanjing, China: Springer, Nov. 2018, pp. 263–277.

[22] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *Proc. 19th IEEE Int. Conf. Mobile Data Manage. (MDM)*, Aalborg, Denmark, Jun. 2018, pp. 125–134.

[23] Y. Zhang, N. Ning, P. Zhou, and B. Wu, "UT-ATD: Universal transformer for anomalous trajectory detection by embedding trajectory information," in *Proc. DMSVIVA*, Jun. 2021, pp. 70–77.
[24] G. Bouritsas, S. Daveas, A. Danelakis, and S. C. A. Thomopoulos, "Automated real-time anomaly detection in human trajectories using sequence to sequence networks," in *Proc. 16th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Taipei, Taiwan, Sep. 2019, pp. 1–8.
[25] Y. Cheng, B. Wu, L. Song, and C. Shi, "Spatial–temporal recurrent neural network for anomalous trajectories detection," in *Advanced Data Mining and Applications*. Dalian, China: Springer, Nov. 2019, pp. 565–578.
[26] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.
[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–20.
[28] T. Kohonen, *Self-Organizing Maps*, vol. 30. Cham, Switzerland: Springer, 2012.
[29] S. Damilola, "A review of unsupervised artificial neural networks with applications," *Int. J. Comput. Appl.*, vol. 181, no. 40, pp. 22–26, Feb. 2019.
[30] M. A. Saleem, W. Nawaz, Y.-K. Lee, and S. Lee, "Road segment partitioning towards anomalous trajectory detection for surveillance applications," in *Proc. IEEE 14th Int. Conf. Inf. Reuse Integr. (IRI)*, San Francisco, CA, USA, Aug. 2013, pp. 610–617.
[31] N. B. Ghrab, E. Fendri, and M. Hammami, "Abnormal events detection based on trajectory clustering," in *Proc. 13th Int. Conf. Comput. Graph., Imag. Visualizat. (CGiV)*, Beni Mellal, Morocco, Mar. 2016, pp. 301–306.
[32] D. T. Lan and S. Yoon, "Trajectory clustering-based anomaly detection in indoor human movement," *Sensors*, vol. 23, no. 6, p. 3318, 2023.
[33] N. Di Mauro and S. Ferilli, "Unsupervised LSTMs-based learning for anomaly detection in highway traffic data," in *Foundations of Intelligent Systems*. Limassol, Cyprus: Springer, Oct. 2018, pp. 281–290.
[34] A. Alamr and A. Artoli, "Unsupervised transformer-based anomaly detection in ECG signals," *Algorithms*, vol. 16, no. 3, p. 152, Mar. 2023.
[35] S. G. K. Patro and K. K. Sahu, "Normalization: A preprocessing stage," 2015, *arXiv:1503.06462*.
[36] Y. Tao, A. Both, R. I. Silveira, K. Buchin, S. Sijben, R. S. Purves, P. Laube, D. Peng, K. Toohey, and M. Duckham, "A comparative analysis of trajectory similarity measures," *GISci. Remote Sens.*, vol. 58, no. 5, pp. 643–669, Jul. 2021.
[37] E. A. Uriarte and F. D. Martín, "Topology preservation in SOM," *Int. J. Appl. Math. Comput. Sci.*, vol. 1, no. 1, pp. 19–22, 2005.
[38] D. O. Olguin, B. N. Waber, T. Kim, A. Mohan, K. Ara, and A. Pentland, "Sensible organizations: Technology and methodology for automatically measuring organizational behavior," *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 39, no. 1, pp. 43–55, Feb. 2009.
[39] M. Szekér, "Spatio-temporal outlier detection in streaming trajectory data," 2014. [Online]. Available: https://www.semanticscholar.org/search?q=Spatio-temporal-outlier-detection-in-streaming-data&sort=relevance
[40] P. Banerjee, P. Yawalkar, and S. Ranu, "MANTRA: A scalable approach to mining temporally anomalous sub-trajectories," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 1415–1424.
[41] Z. Zhu, D. Yao, J. Huang, H. Li, and J. Bi, "Sub-trajectory-and trajectory-neighbor-based outlier detection over trajectory streams," in *Advances in Knowledge Discovery and Data Mining*, vol. 10937. Cham, Switzerland: Springer, Jun. 2018, pp. 551–563. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-93034-3_44#citeas
[42] Y. Liu, K. Zhao, G. Cong, and Z. Bao, "Online anomalous trajectory detection with deep generative sequence modeling," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Dallas, TX, USA, Apr. 2020, pp. 949–960.
[43] P.-R. Lei, "A framework for anomaly detection in maritime trajectory behavior," *Knowl. Inf. Syst.*, vol. 47, no. 1, pp. 189–214, Apr. 2016.
[44] P. Han, A. L. Ellefsen, G. Li, F. T. Holmeset, and H. Zhang, "Fault detection with LSTM-based variational autoencoder for maritime components," *IEEE Sensors J.*, vol. 21, no. 19, pp. 21903–21912, Oct. 2021.

DOI THI LAN received the B.S. and M.S. degrees in electrical and electronic engineering from Le Quy Don Technical University, Hanoi, Vietnam, in 2013 and 2018, respectively. She is currently pursuing the Ph.D. degree in electrical, electronic and computer engineering with the University of Ulsan, South Korea. Her research interests include applying machine learning to anomalous trajectory detection and human mobility prediction.

SEOKHOON YOON (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science and engineering from the State University of New York at Buffalo (SUNY Buffalo), in 2005 and 2009, respectively. He was a senior research engineer with defense industry, where he designed several tactical wireless network solutions. He is currently a Professor with the University of Ulsan, South Korea, where he leads the Advanced Mobile Networks and Intelligent Systems Laboratory. His research interests include opportunistic networking, human mobility, intelligence-defined networking, and machine learning-based IoT services.