SiamCircle

Trajectory Representation Learning in Free Settings

Nasri, Maedeh; Baratchi, Mitra; Koutamanis, Alexander; Rieffe, Carolien

DOI
10.1007/978-3-031-91398-3_6

Publication date
2025

Document Version
Final published version

Published in
Advances in Intelligent Data Analysis XXIII

Citation (APA)
Nasri, M., Baratchi, M., Koutamanis, A., & Rieffe, C. (2025). SiamCircle: Trajectory Representation Learning in Free Settings. In G. Krempl, K. Puolamäki, & I. Miliou (Eds.), *Advances in Intelligent Data Analysis XXIII: 23rd International Symposium on Intelligent Data Analysis, IDA 2025, Konstanz, Germany, May 7–9, 2025, Proceedings* (pp. 67-80). (Lecture Notes in Computer Science; Vol. 15669 LNCS). Springer. https://doi.org/10.1007/978-3-031-91398-3_6

SiamCircle: Trajectory Representation Learning in Free Settings

Maedeh Nasri[1], Mitra Baratchi[2], Alexander Koutamanis[3], and Carolien Rieffe[1,4,5(✉)]

[1] Department of Developmental Psychology, Leiden University, Leiden, The Netherlands
[2] Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands
[3] Faculty of Architecture and the Built Environment, Delft University of Technology, Delft, The Netherlands
[4] Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands
c.j.rieffe@utwente.nl
[5] Department of Psychology and Human Development, Institute of Education, UCL, London, UK

Abstract. Trajectory representation learning (TRL) is an intermediate step in handling trajectory data to realize various downstream machine-learning tasks. While most previous TRL research focuses on modeling structured movements in large-scale urban spaces (e.g., cars or pedestrians on streets), this paper focuses on a more challenging scenario of modeling free movement in small-scale social spaces (e.g., children playing in a schoolyard). We present a TRL model, SiamCircle, to process raw trajectories without additional feature extraction to prevent information loss. SiamCircle adopts a Siamese network with Circle Loss to learn trajectory embeddings. Furthermore, SiamCircle employs a data augmentation process to enable self-supervised learning and enrich the input data to address the limited access to high-quality data and ground truth. We evaluate the performance of SiamCircle in downstream tasks using trajectory ranking and clustering performance via seven evaluation metrics collectively. Using an ablation study, we explored the impact of different loss functions on the model's performance. Accordingly, we selected a 2-D convolutional design with Circle Loss as the best-performing model. In a comparative study, we compared our model against three other baselines. We observed up to 19% improvements in trajectory ranking tasks and achieved the highest average rank in supervised clustering tasks.

Keywords: Triplet Loss · Clustering · Trajectory Representation Learning

1 Introduction

In the wake of rapid growth in wearable technologies, people generate large amounts of movement data using location-aware devices like sports bands and

G. Krempl et al. (Eds.): IDA 2025, LNCS 15669, pp. 67–80, 2025.
https://doi.org/10.1007/978-3-031-91398-3_6

smartwatches. Various organizations address existing challenges in, for example, traffic forecasting [26] and animal migration patterns [13], by collecting and analyzing this data, known as spatio-temporal movement data or trajectories [6].

In a broader context, this movement data is captured in mainly two settings: *structured settings* and *free settings*. In structured settings, movement is recorded in areas influenced by spatial features or regulations, e.g., road networks or driving policies. This setting often exhibits periodic patterns, such as commuting to work on weekdays. Data from structured settings are valuable for applications like traffic forecasting. Conversely, in free settings, individuals move without restrictions, resulting in unstructured trajectories, which often occur in constrained environments, such as students' movements in a schoolyard or athletes' movements in sportsfields. This data is useful for analyzing micro-level behaviors, including social interactions and group dynamics.

In the context of trajectory analysis in structured settings, trajectory representation learning (TRL) has attracted extensive research attention in various machine learning tasks such as trajectory flow forecasting [6] or pair-wise similarity computation [4,27]. Yet, one largely unsolved challenge is effectively designing a TRL framework for movements occurring in free settings. Developing TRL models in free settings is much more challenging than in structured settings due to (i) irregular movement patterns and movements formed in the absence of typical urban features, (ii) limitations in data acquisition systems leading to imperfect location data and complicating trajectory analysis, and (iii) privacy concerns restricting the collection and sharing of high-quality data.

Recent studies in modeling movements in structured settings implement preprocessing methods such as grid-cell projection [15] and graph embeddings [4]. While the results are promising, their application to trajectories collected in free settings may be limited due to the potential information loss during the pre-processing. Besides, their performance highly relies on the choice of hyperparameters, e.g., cell size. In a free setting, finding the optimal grid size is more challenging, especially in the presence of imprecisions due to noise.

This paper presents a robust trajectory representation learning framework called SiamCircle, which directly applies to raw trajectory data collected in free movement settings. SiamCircle consists of three main components: (1) a data augmentation process to generate the augmented trajectories accounting for imperfections in trajectories and enriching data sets with limited sample size, (2) a neural network applicable to raw trajectory data that maximally exploits spatial and temporal similarities among similar trajectories while amplifying differences between non-similar ones, eliminating common information loss during feature extraction, (3) a loss function uniquely designed with our learning framework to learn representative features through the training process. To our knowledge, this is the first study proposing a TRL framework using raw trajectories collected in free settings. Overall, this paper makes the following contributions:

– We develop a framework to create augmented trajectories that simulate similar movements and imperfections in data. Moreover, this framework generates a ground-truth dataset that annotates the trajectories based on their simi-

larity class and enriches datasets with a limited sample size. Such annotated dataset can be used in supervised training and supervised evaluation metrics.
– We propose a unique deep neural network model with triplet-based Circle Loss to learn trajectory representations directly from raw trajectories to limit information loss commonly occurring during the feature extraction.
– We demonstrate the performance of the proposed framework by conducting an ablation study and a comparative study on five trajectory datasets collected in a free setting using seven evaluation metrics in downstream tasks using two classes of metrics, namely trajectory ranking, and clustering. We compare the performance of SiamCircle to three other baseline models.

The remaining part of the paper is organized as follows. In Sect. 2, we discuss related work. Section 3 presents the problem statement, and Sect. 4 explains the details of our proposed method. The experimental setup and results are presented in Sect. 5 and Sect. 6, respectively. Finally, we summarize the paper and discuss future research directions in Sect. 7.

2 Related Work

Several studies proposed TRL models primarily in structured settings like urban traffic forecasting and detecting transportation modes [3,11]. While these models show promise, they become impractical for data collected in free settings, especially in the absence of external information.

Moreover, TRL via Contrastive learning has been employed in various machine learning problems in natural language processing [10], image processing [21], and spatio-temporal problems [4,7]. Contrastive learning is especially beneficial when using noisy and imperfect datasets with limited sample sizes. Fan et al. [7] proposes a contrastive learning approach powered by Triplet loss [20] in combination with a sliced encoder network, S-BiLSTM, for a user re-identification problem based on trajectory data. S-BiLSTM learns the robust part of individuals' trajectories (segmented in 24 h). This segmentation is not applicable for movements in free settings where trajectories do not exhibit daily patterns. TrajCL [4] is another contrastive learning model including a dual-feature multi-head self-attention-based encoder with InfoNCE loss [17] using trajectories in urban areas. TrajCL proposed a pointwise trajectory feature enrichment method to extract structural and point features to train their model. The optimization process used in both loss functions, Triplet loss and InfoNCE loss, is rigid, as their loss calculations give pairs with different similarities an equal pre-defined margin.

The present study uses a Siamese network with Circle Loss to learn trajectories' representative features designed explicitly for free movements in constrained environments (i.e., free settings). Our model uses raw trajectories without any feature extraction techniques as opposed to earlier work [4,28] to limit the unnecessary loss of data and oversimplification in free settings. Circle Loss adopts a dynamic penalty scheme based on the degree of similarity between trajectories in a triplet to obtain a sustainable optimization process and high-quality embeddings to address the rigidness problem in earlier works [4,28].

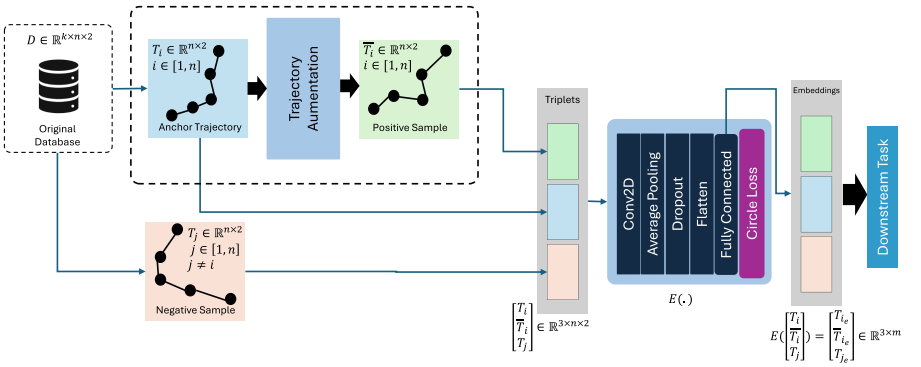

Fig. 1. An overview of SiamCircle: Data Augmentation Process to create the [Anchor, Positive, and Negative] triplets, Siamese Network with Circle Loss to generate the embeddings of the given raw triplets.

3 Problem Definition

Our trajectory modeling problem is based on a given trajectory dataset D, where each trajectory $\{T_i(\mathbf{x}, \mathbf{y}) \in D | (\mathbf{x}, \mathbf{y}) = \{(x_1, y_1), \cdots, (x_n, y_n)\}\}$ is a sequence of two-dimensional points recording the trace of the moving object i over n timestamps.

We are interested in learning an embedding function $E(\cdot)$ that represents trajectory T_i as $E(T_i) = T_{i_e} \in \mathbb{R}^m$ ($m \ll n$ is the dimension of embedding space) such that $f(T_{j_e}, T_{i_e})$ is minimized for any pair of the object i and j moving along the same underlying route, and, conversely, maximized for any pair of objects i and j moving along different underlying routes. While $f(\cdot, \cdot)$ is a distance function, e.g., Euclidean distance.

4 Methodology

This section discusses our proposed method for TRL. First, we explain the trajectory augmentation process. Next, the neural network design is presented. Lastly, we present the loss function used in the proposed framework. An overview of our method is presented in Fig. 1.

4.1 Trajectory Augmentation

In this section, we propose a trajectory augmentation method that adopts meaningful transformations to generate augmented trajectories from original ones. Previously proposed trajectory augmentation methods involve applying predefined transformations to the original data, for instance, by truncating or point masking [4,15]. These transformations often modify the trajectory length, making them only applicable in combination with manual feature extraction methods to create fixed-length sequential data for neural network models. However,

as we aim to utilize raw trajectories with no feature extraction, we propose a unique trajectory augmentation approach where the trajectory length remains unchanged while meaningful transformations are applied. Our trajectory augmentation process serves three main purposes: (1) simulating similar movement trajectories by distorting original trajectories in different scales (i.e., additive noise) to account for different levels of similarities, (2) enriching limited-size datasets by adding augmented trajectories to the original dataset used purely for training. Thus enabling a self-supervised training process for the neural network model, and (3) simulating the practical imperfections found in location data acquisition systems in the form of large-scale noise (i.e., large spatio-temporal displacements or jumps), aiming to train a model that is robust to issues such as noise and other imperfections in the data. To achieve these goals, the augmented trajectory $\bar{T}_i$ is created by applying pre-defined transformations to each original trajectory T_i of a moving object i, as formulated in Eq. 1:

$$\bar{T}_i(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = (\mathbf{x} + \mathbf{x}'_i, \mathbf{y} + \mathbf{y}'_i), \qquad \begin{cases} \mathbf{x}'_i = \mathcal{U}(l_x, h_x).d_s, \\ \mathbf{y}'_i = \mathcal{U}(l_y, h_y).d_s, \end{cases} \tag{1}$$

where $\bar{T}_i(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is the augmented trajectory of moving object i. $\mathbf{x}'_i$ is the coordinate of the original trajectory T_i. $\mathbf{x}'_i$ is the distortion vector across $\mathbf{x}$. $\mathcal{U}(l_x, h_x)$ is the uniform distribution bounded between lower bound $l_x = -\sigma(\Delta\mathbf{x})$ and higher bound $h_x = \sigma(\Delta\mathbf{x})$, in which $\sigma(.)$ is the standard deviation over $\Delta\mathbf{x}$. and $\Delta\mathbf{x}$ is the vector of positional differences across $\mathbf{x}$ coordinates. d_s is a distortion scale randomly selected among the given scaling range. In all the mentioned annotations, replacing $\mathbf{x}$ with $\mathbf{y}$ gives the same definition across the $\mathbf{y}$ coordinate (instead of $\mathbf{x}$). In our experiments, we acquire a single augmented trajectory by applying two types of distortions namely *additive noise* and *jumps* (by setting two values for d_s). While additive noise applies to all coordinates of the original data, jumps only apply randomly to k number of samples. Another difference between additive noise and jumps is the scale of the distortion. In the additive noise, d_s includes a lower distortion scale to create similar movements, satisfying aims (1) and (2). In the jumps, d_s includes a larger distortion scale to create imperfections in trajectories, satisfying aims (2) and (3). Thus, the semi-synthesized trajectory, $\bar{T}_i(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, creates one variation of the object i in the database per distortion scale. In Sect. 4.2, we will discuss our proposed neural network model.

4.2 Siamese Network Architecture

This section discusses the design of our proposed neural network model. The unique aspect of this design is the model's ability to be directly applied to raw 2-D trajectories without implementing any preliminary pre-processing steps. The rationale behind this choice is to include as much information as possible without abstracting spatial details, e.g., trajectory griding, or extracting spatio-temporal features, e.g., speed. To achieve this goal, we adopted a Triplet-based Siamese network, which typically consists of three identical networks with shared

weights [2]. The primary purpose of Siamese networks is to learn the similarity between inputs by comparing their representations. In this design, the model takes an anchor trajectory together with a positive sample (i.e., a trajectory from the same class as the anchor) and a negative sample (i.e., a trajectory from a different class than the anchor) as a triplet input, pass each sample through its identical branch, and hand over the generated embeddings to a triplet-based loss function. Specifically, our model contains two sections:

2D-CNN/Average Pooling/Dropout. Inspired by the use of neural networks in time-series forecasting [25], this section employs a 2D convolutional layer (2D-CNN) as the initial feature extraction step. The 2D-CNN is particularly effective in capturing spatial-temporal dependencies within trajectory data, as it slides learnable filters over the input to identify local patterns.

Following the 2D-CNN, an average pooling layer is introduced to reduce dimensionality and filter out noise from the extracted feature maps. This pooling mechanism computes the average values within a small window, which helps the network generalize better by emphasizing essential trajectory trends while discarding less significant details.

To further enhance model robustness, a dropout layer is applied after pooling. Dropout randomly deactivates a subset of neurons during training, which prevents the network from overfitting. This regularization technique ensures that the model remains adaptable and performs well on unseen trajectory data.

In summary, this section of the framework transforms raw trajectory data into a compact yet informative representation by extracting relevant spatial-temporal patterns, filtering out noise, and preventing overfitting.

Fully-Connected Layer. The outputs from the previous layer representing high-level embedding features are fed to a fully connected (FC) layer. This allows the model to learn the non-linear combinations of these features. This FC layer serves as a crucial component in transforming the extracted spatial-temporal features into a more discriminative representation suitable for trajectory ranking and clustering tasks. The generated embeddings will be given to Circle Loss as detailed in Sect. 4.3.

4.3 Loss Function: Circle Loss

Most loss functions used for measuring similarity, including the triplet loss [20], directly incorporate the pairwise distances between trajectory pairs into a similarity score. This optimization approach is rigid as the similarity score linearly impacts the loss. Ideally, we would like to give greater emphasis (or penalty) when a similarity score significantly deviates from the optimum. To this end, Sun. et al. [23] proposed Circle Loss, which re-weights each similarity to highlight the less optimized similarity scores. The Circle Loss was initially proposed in computer vision with a unified formula for two elemental deep feature learning paradigms, i.e., learning with class-level and pair-wise labels, formulated as follows:

$$L = log\left[1 + \sum_{j=1}^{L} exp\left(\gamma\alpha_n^j(s_n^j - \Delta_n)\right)\sum_{i=1}^{K} exp\left(-\gamma\alpha_p^i(s_p^i - \Delta_p)\right)\right], \begin{cases}\alpha_p^i = [O_p - s_p^i]_+, \\ \alpha_n^j = [s_n^j - O_n]_+, \end{cases} \quad (2)$$

in which n and p annotations in a pairwise label paradigm refer to dissimilar and similar pairs. γ is the scale factor. L and K are the number of dissimilar and similar samples. Δ_n and Δ_p are dissimilar and similar margins, and α_n^j and α_p^i are non-negative weighting factors for pairwise distances between dissimilar samples (i.e., s_n) and similar samples (i.e., s_p), respectively. In their calculation, $[\cdot]_+$ is the "cut-off at zero" operation to ensure α_p^i and α_n^j are non-negative. O_p and O_n is the optimum similarity score for s_p and s_n respectively. To reduce the number of hyperparameters, $O_p = 1 + m$, $O_n = -m$, $\Delta_p = 1 - m$, and $\Delta_n = m$. Hence, there are only two hyper-parameters, i.e., the scale factor γ and the relaxation margin m. In a class-level paradigm, the dissimilar and similar terminologies are replaced with between-class and within-class terms using the same equation as Eq. 2. Analytically, Circle Loss offers a more flexible optimization approach towards a more definite convergence target.

In our proposed model, Circle Loss is adopted for the first time in TRL via a triplet learning paradigm using a Siamese network. We collect an equal number of dissimilar and similar pairs in each batch (i.e., $L = K$, equal to a pre-defined batch size) to create a triplet of anchor, positive, and negative samples. This also creates a balanced batch (with an equal number of positive and negative samples) in each epoch to prevent overfitting and bias towards a particular class. Moreover, we adopted the "Hard" triplet strategy [20]. In comparison with the random triplet strategy, where triplets are selected randomly, applying this strategy disregards triplets that are too easy, thereby reducing the risk of overfitting.

5 Experiments

We evaluate SiamCircle on five real trajectory datasets using seven evaluation metrics in trajectory ranking and clustering tasks. The following sections describe the experimental settings, datasets, baselines, and evaluation metrics.

Experimental Settings. SiamCircle is implemented in Tensorflow. The details of the implementation is provided in the Github repository.[1] We report the average and standard deviation of results across ten runs for each experiment. The Wilcoxon signed rank test [24] has been applied to investigate the significant differences between the top two performing models and to rank algorithms based on their performances in different metrics. In the following sections, we present the details of datasets, baselines, and evaluation metrics used in our study.

[1] The code is available at https://anonymous.4open.science/r/SiamCircle-Trajectory-Representation-Learning-17C3/.

Table 1. The characteristics of Opentraj datasets. The columns indicate the name of the dataset, the captured area, and the data size for Train, augmented train (i.e., Train($\overline{T}$)) and Test splits in the form of *(Sample Size, Number of groups).

Datasets	Area	Train (T)	Train($\overline{T}$)	Test
Eth	224.704	(40, 15)	(5098, 15)	(96, 37)
Hotel	67.210	(12, 6)	(5027, 6)	(31, 15)
Zara01	171.504	(34, 16)	(5050, 16)	(80, 35)
Zara02	158.063	(71, 35)	(5164, 35)	(167, 82)
Student03	242.884	(185, 61)	(5242, 61)	(433, 194)

Datasets. Five pedestrian datasets -*eth*, *hotel* [18], *zara01*, *zara02*, and *students03* [14]-all captured in a free setting-are utilized in the experiments. These are widely recognized benchmarks for group detection tasks using spatio-temporal data [1]. They contain the location and velocity of movements across multiple timeframes, including ground truth information on group membership. Table 1 shows the main characteristics of these datasets. Since the number of samples per dataset is limited, we adopted our novel data augmentation process described in Sect. 4.1 to generate augmented trajectories per class (or group).

Baseline. We compared SiamCircle with three other deep learning models, namely S-BiLSTM [7], TrajCL [4], and Trajectory2vec [28]. We use the released code and default parameters for all baseline methods except S-BiLSTM, which has no released code. We implement this method following their original study [7]. In TrajCL, the cell size is set to 5 as it performed better than the default value, i.e., cell size = 100, as it was designed for city-scale datasets.

Evaluation Metrics. We have adopted seven evaluation metrics in trajectory ranking and clustering tasks. The trajectory ranking task is evaluated by the top-k hitting ratio, while the clustering task is evaluated by two supervised metrics and two unsupervised metrics as follows:

Top-K Hitting Ratio. This metric examines the overlap of the *top-k*, for $k = 1, 5$, and 10, k, sorted distances between embeddings and the ground truth. Specifically, we use the Euclidean measure to calculate the distances between the original trajectory T and other trajectories in the test set. Analogously, we calculate the Euclidean distances between $E(T)$ and the embeddings of other trajectories in the test set. We then sort the obtained distances in both sets and find the number of common indexes across the window size of k. The higher overlap ratio shows better effectiveness of the measure.

Clustering Performance. Retrieving the embedding of raw trajectories allows a clustering algorithm to better identify groups of similar trajectories. To test this,

we selected two supervised evaluation metrics, i.e., Normalized Mutual Information (NMI) [22] and Fowlkes Mallows (FM) score [8], and two unsupervised metrics, i.e., Davies Bouldin (DB) score [5] and Silhouette (Si) [19], to evaluate the performance of a K-mean Clustering Algorithm [16]. The number of ground truth groups, as indicated in Table 1, defines the number of clusters in the K-mean Algorithm. In summary, the NMI score measures the agreement of the two assignments, with the highest score being 1. The FM score is the geometric mean of the pairwise precision and recall, rating from 0 to 1. The Si score indicates how well clusters are separated from each other, ranging between -1 and 1. DB score is the average similarity measure of each cluster with its most similar cluster, where the minimum score is zero, with lower values indicating better clustering. In all other metrics, the higher score indicates better-defined clusters.

6 Results

In this section, the experiment's results are presented in two studies: (i) an ablation study to explore different design options and (ii) a comparative study to compare the performance of our proposed model against three baselines.

Ablation Study. Our ablation study includes loss function analysis to investigate the impact of different loss functions in our framework. Thus, we trained our proposed neural network design, i.e., $1 \times Conv2D$, with Triplet loss [20], Contrastive loss [12], USR loss [9], and Circle Loss. As shown in Table 2, the model $1 \times Conv2D$ with Circle Loss, on average, ranked higher than all the other models in almost all metrics across all datasets. There is only one exception on the unsupervised clustering task where $1 \times Conv2D$ with Circle Loss ranked last in average rank (M_R) and the rank of each unsupervised metric (i.e., DB and Si). A deeper analysis of the data shows that the worst performance of these two measures is in the $Hotel$ dataset, which includes the smallest area with the smallest sample size in the original datasets (See Table 1). This makes the clusters less distinguishable, which might explain the poor performance of these two metrics, as both indicate how well clusters are separated. Yet, the other loss functions did not consistently outperform in either of these measures. Hence, $1 \times Conv2D$ with Circle Loss, i.e., SiamCircle, is selected and used in the following sections to conduct comparative experiments.

Comparative Study. In this section, we compare the performance of our proposed model against three baselines, namely S-BiLSTM, TrajCL, and Trajectory2vec, in trajectory ranking and clustering tasks using the seven evaluation metrics. The results are depicted in Table 3.

Trajectory Ranking. SiamCircle performed significantly higher than the baselines in the top Hit-k measure in almost all cases except in the Hit-1 metric of the

Table 2. The result of the ablation study, in which the highest ranks in each evaluation metric are shown in Boldface. R and M_R denote the performance ranking and the average rank per category. Horizontal lines separate categories.

Loss Function	Metric	Datasets					R	M_R
		ETH	Hotel	Student03	Zara01	Zara02		
Triplet loss	Hit-1	0.76	0.677	0.479	0.6	0.669	1.8	1.9
	Hit-5	0.79	0.852	0.67	0.758	0.801	2.0	
	Hit-10	0.857	0.935	0.69	0.814	0.815	2.0	
	NMI	0.869	0.934	0.864	0.774	0.836	2.4	2.3
	FM	0.406	0.765	0.174	0.123	0.063	2.2	
	DB	0.669	0.588	0.504	0.521	0.288	2.7	2.7
	Si	0.345	0.434	0.322	0.286	0.334	2.8	
Contrastive loss	Hit-1	0.49	0.613	0.407	0.312	0.585	2.8	2.9
	Hit-5	0.623	0.761	0.544	0.503	0.754	3.0	
	Hit-10	0.749	0.848	0.587	0.585	0.77	3.0	
	NMI	0.79	0.845	0.87	0.788	0.844	2.6	2.6
	FM	0.227	0.436	0.212	0.164	0.07	2.6	
	DB	0.606	0.336	0.454	0.643	0.239	2.0	**2.0**
	Si	0.395	0.38	0.34	0.253	0.394	**2.0**	
USR loss	Hit-1	0.302	0.452	0.172	0.162	0.296	4.0	4.0
	Hit-5	0.331	0.742	0.193	0.268	0.337	4.0	
	Hit-10	0.424	0.806	0.229	0.35	0.352	4.0	
	NMI	0.801	0.82	0.858	0.751	0.838	3.6	3.7
	FM	0.248	0.34	0.152	0.056	0.049	3.8	
	DB	0.548	0.22	0.504	0.305	0.293	**1.7**	**2.0**
	Si	0.392	0.595	0.301	0.479	0.285	2.4	
Circle Loss	Hit-1	0.771	0.523	0.686	0.724	0.732	**1.4**	**1.1**
	Hit-5	0.84	0.88	0.738	0.791	0.909	**1.0**	
	Hit-10	0.892	0.953	0.744	0.843	0.877	**1.0**	
	NMI	0.858	0.878	0.911	0.811	0.849	**1.4**	**1.4**
	FM	0.39	0.517	0.399	0.223	0.102	**1.4**	
	DB	0.627	0.671	0.526	0.589	0.324	3.6	3.2
	Si	0.328	0.218	0.331	0.287	0.366	2.8	

Hotel dataset. However, the highest performance (obtained by S-BiLSTM) is not statistically significant. Moreover, the SiamCircle outperformed with an average performance gap of 19% compared with the second best-performing model, i.e., BiLSTM. This shows that the embedding vectors retrieved by our model are helpful for trajectory ranking tasks to estimate the top similar trajectories.

Table 3. The comparative study result (mean±standard division). The statistically significant results compared to the second best results are shown by *. The highest ranks per evaluation metric are shown in Boldface. R and M_R denote the performance ranking and the average rank per category.

Model	Metric	Datasets					R	M_R
		ETH	Hotel	Student03	Zara01	Zara02		
S-BiLSTM	Hit-1	0.449±0.497	0.561±0.496	0.42±0.494	0.34±0.474	0.554±0.497	1.8	1.9
	Hit-5	0.651±0.209	0.78±0.187	0.582±0.236	0.488±0.215	0.661±0.239	2.0	
	Hit-10	0.699±0.176	0.854±0.15	0.614±0.204	0.587±0.145	0.687±0.228	2.0	
	NMI	0.799±0.01	0.859±0.036	0.864±0.006	0.763±0.014	0.851±0.009	**1.8**	2.1
	FM	0.245±0.025	0.455±0.103	0.188±0.021	0.124±0.034	0.1±0.029	2.4	
	DB	0.474±0.023	0.391±0.041	0.466±0.023	0.39±0.048	0.281±0.021	1.6	1.9
	Si	0.415±0.014	0.4±0.05	0.31±0.012	0.323±0.032	0.388±0.019	2.2	
TrajCL	Hit-1	0.097±0.296	0.106±0.308	0.071±0.257	0.094±0.291	0.079±0.27	3.8	3.5
	Hit-5	0.194±0.188	0.476±0.323	0.182±0.209	0.204±0.179	0.192±0.216	3.4	
	Hit-10	0.288±0.19	0.566±0.167	0.234±0.172	0.32±0.167	0.276±0.181	3.2	
	NMI	0.731±0.01	0.754±0.015	0.836±0.005	0.756±0.008	0.782±0.002	4.0	3.8
	FM	0.158±0.024	0.171±0.055	0.136±0.014	0.126±0.024	0.068±0.002	3.6	
	DB	0.486±0.069	0.422±0.102	**0.405 ± 0.024***	0.335±0.108	**0.073 ± 0.022***	**1.4**	**1.2**
	Si	0.435±0.056	0.412±0.082	**0.436 ± 0.048***	**0.438 ± 0.066***	**0.518 ± 0.007***	**1.0**	
Trajectory2vec	Hit-1	0.198±0.398	0.319±0.466	0.063±0.243	0.154±0.361	0.275±0.446	3.2	3.5
	Hit-5	0.134±0.142	0.401±0.272	0.096±0.131	0.23±0.237	0.376±0.304	3.6	
	Hit-10	0.16±0.107	0.455±0.141	0.117±0.11	0.268±0.135	0.385±0.291	3.8	
	NMI	0.775±0.008	0.843±0.018	0.841±0.003	0.807±0.008	0.85±0.003	2.4	2.4
	FM	0.189±0.017	0.416±0.064	0.08±0.009	0.237±0.034	**0.149 ± 0.01***	2.4	
	DB	0.638±0.028	0.472±0.028	0.611±0.013	0.564±0.035	0.31±0.014	3.2	3.5
	Si	0.277±0.011	0.388±0.028	0.221±0.006	0.262±0.014	0.324±0.009	3.8	
SiamCircle	Hit-1	**0.771 ± 0.420***	0.523±0.449	**0.686 ± 0.464***	**0.724 ± 0.447***	**0.732 ± 0.443***	**1.2**	**1.0**
	Hit-5	**0.840 ± 0.144***	**0.880 ± 0.118***	**0.738 ± 0.195***	**0.791 ± 0.146***	**0.909 ± 0.116***	**1.0**	
	Hit-10	**0.892 ± 0.086***	**0.953 ± 0.055***	**0.744 ± 0.155***	**0.843 ± 0.1***	**0.877 ± 0.146***	**1.0**	
	NMI	**0.849 ± 0.009***	0.858±0.026	**0.921 ± 0.004***	**0.805 ± 0.007***	0.838±0.003	**1.8**	**1.7**
	FM	**0.366 ± 0.025***	0.458±0.083	**0.451 ± 0.019***	0.213±0.018	0.07±0.007	**1.6**	
	DB	0.668±0.025	0.618±0.046	0.529±0.015	0.584±0.023	0.328±0.012	3.8	3.4
	Si	0.329±0.012	0.257±0.024	0.331±0.007	0.289±0.011	0.379±0.006	3.0	

Clustering. SiamCircle performed higher than the other baselines in supervised clustering measures, i.e., NMI and FM scores (with S-BiLSTM performing similarly to SiamCircle in the NMI metric), which was in contrast with the result of unsupervised clustering measures, i.e., DB and Si. This could be due to providing explicit positive and negative samples for training in these two models, which allows the models to learn more specific patterns and relationships. Additionally, the dynamic penalty strategy by Circle Loss likely helped SiamCircle achieve a higher rank than S-BiLSTM. On the other hand, TrajCL has ranked best in unsupervised clustering metrics, likely because TrajCL trains without using group membership. Thus, the obtained embeddings via TrajCL form clusters in the embedding space that do not necessarily reflect individual group memberships. This makes TrajCL particularly useful when ground truth is not

available. Yet, a deep understanding of data is required to translate the findings into an implication.

Overall, SiamCircle has ranked highest compared to the baselines in Trajectory Ranking. In Trajectory Clustering, the highest performance is obtained only in supervised metrics. This demonstrates the strong capability of representations obtained by SiamCircle to be used in various domains and applications.

7 Conclusion

This study revisits the problem of TRL, specifically when trajectories are collected from free movements in small areas. We introduced a novel framework, SiamCircle, which includes a Siamese framework with Circle Loss. We conducted experiments with five benchmark datasets, using seven evaluation metrics in trajectory ranking and clustering tasks. In the ablation study, we demonstrated that one 2-dimensional convolutional layer together with Circle Loss, on average, outperformed other candidates. In a comparative study, SiamCircle consistently outperformed other models in trajectory ranking with an average performance gap of 19% compared with the second best-performing model, i.e., BiLSTM, and in unsupervised clustering measures. In future research, automatically setting the parameter of Circle loss based on the characteristics of the given datasets can be explored to make it more adaptable to diverse datasets without requiring manual hyperparameter adjustments which improves its usability in real-world applications. SiamCircle can be used in various tasks, such as analyzing social interactions in free settings, e.g., schoolyards and sports clubs. This opens up possibilities for behavioral studies, crowd analysis, and human-computer interaction research.

References

1. Amirian, J., Zhang, B., Castro, F.V., Baldelomar, J.J., Hayet, J.B., Pettré, J.: Opentraj: assessing prediction complexity in human trajectories datasets. In: Proceedings of the Asian Conference on Computer Vision (2020)
2. Bromley, J., et al.: Signature verification using a "siamese" time delay neural network. Int. J. Pattern Recogn. Artif. Intell. **7**(04) (1993)
3. Cardoso-Pereira, I., Borges, J.B., Viana, A.C., Loureiro, A.A., Ramos, H.S.: Popayi: muscling ordinal patterns for low-complex and usability-aware transportation mode detection. IEEE Internet Things J. (2024)
4. Chang, Y., Qi, J., Liang, Y., Tanin, E.: Contrastive trajectory similarity learning with dual-feature attention. In: 2023 IEEE 39th International Conference on Data Engineering (ICDE), pp. 2933–2945. IEEE (2023)
5. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **2**, 224–227 (1979)
6. Ermagun, A., Levinson, D.: Spatiotemporal traffic forecasting: review and proposed directions. Transp. Rev. **38**(6), 786–814 (2018)
7. Fan, Z., Song, X., Chen, Q., Jiang, R., Shibasaki, R., Tsubouchi, K.: Trajectory fingerprint: one-shot human trajectory identification using siamese network. CCF Trans. Perv. Comput. Interact. **2**(2), 113–125 (2020)

8. Fowlkes, E.B., Mallows, C.L.: A method for comparing two hierarchical clusterings. J. Am. Stat. Assoc. **78**(383), 553–569 (1983)
9. Franceschi, J.Y., Dieuleveut, A., Jaggi, M.: Unsupervised scalable representation learning for multivariate time series. Adv. Neural Inf. Process. Syst. **32** (2019)
10. Jiang, C., et al.: Hallucination augmented contrastive learning for multimodal large language model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 27036–27046 (2024)
11. Ju, W., et al.: Cool: a conjoint perspective on spatio-temporal graph neural network for traffic forecasting. Inf. Fusion **107**, 102341 (2024)
12. Khosla, P., et al.: Supervised contrastive learning. Adv. Neural. Inf. Process. Syst. **33**, 18661–18673 (2020)
13. Kudari, M., Nandeppanavar, A.S., Thotad, P., Jatti, S., Koti, S.: A machine learning based approach for bird migration detection and feature analysis with shap. In: 2024 IEEE North Karnataka Subsection Flagship International Conference (NKCon), pp. 1–7. IEEE (2024)
14. Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example. In: Computer Graphics Forum, vol. 26, pp. 655–664. Wiley Online Library (2007)
15. Li, X., Zhao, K., Cong, G., Jensen, C.S., Wei, W.: Deep representation learning for trajectory similarity computation. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE), pp. 617–628. IEEE (2018)
16. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, vol. 1, pp. 281–297 (1967)
17. Oord, A.V.D., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
18. Pellegrini, S., Ess, A., Schindler, K., Van Gool, L.: You'll never walk alone: modeling social behavior for multi-target tracking. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 261–268. IEEE (2009)
19. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20** (1987)
20. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)
21. Shi, J., Gao, P., Qin, J.: Transformer-based no-reference image quality assessment via supervised contrastive learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 4829–4837 (2024)
22. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res. **3**(Dec) (2002)
23. Sun, Y., et al.: Circle loss: a unified perspective of pair similarity optimization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020)
24. Wilcoxon, F.: Individual comparisons by ranking methods. Biometr. Bull. **1**(6), 80–83 (1945)
25. Yan, A., Cheng, S., Kang, W.C., Wan, M., McAuley, J.: Cosrec: 2d convolutional neural networks for sequential recommendation. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management (2019)
26. Yang, B., Fan, F., Ni, R., Wang, H., Jafaripournimchahi, A., Hu, H.: A multi-task learning network with a collision-aware graph transformer for traffic-agents trajectory prediction. IEEE Trans. Intell. Transp. Syst. **25**(7), 6677–6690 (2024)

27. Yang, P., Wang, H., Yang, J., Qian, Z., Zhang, Y., Lin, X.: Deep learning approaches for similarity computation: a survey. IEEE Trans. Knowl. Data Eng. (2024)
28. Yao, D., Zhang, C., Zhu, Z., Huang, J., Bi, J.: Trajectory clustering via deep representation learning. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 3880–3887. IEEE (2017)