A compass towards equity: a data analysis framework to capture children's behaviour in the playground context

Nasri, M.

CHAPTER 5

SiamCircle: Trajectory Representation Learning in Free Settings

The contents of this chapter are based on the following submitted study:

- M. Nasri, M. Baratchi, A. Koutamanis, C. Rieffe, "SiamCircle: Trajectory Representation Learning in Free Settings", IDA, 2025.

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 SiamCircle, a novel contrastive learning approach for TRL uniquely designed 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 three classes of metrics, namely trajectory ranking, trajectory similarity, and clustering performance, using collectively nine evaluation metrics. Using an ablation study, we explored the impact of different neural network components and 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 trajectory similarity and supervised clustering tasks.

5.1 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 smartwatches. Various organizations address existing challenges by collecting and analyzing this data, known as spatio-temporal movement data or trajectories [192].

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 sports. This data is useful for analyzing micro-level behaviors, including social interactions and

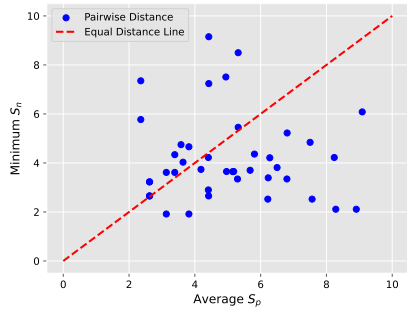

Figure 5.1: Average pairwise distances between a trajectory and trajectories from the same group (S_p) versus from different groups (S_n). Many trajectories have smaller S_n than their S_p, i.e., some samples are positioned below the red line.

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 [192] or pair-wise similarity computation [193]. Yet, one largely unsolved challenge is effectively designing a TRL framework for movements occurring in free settings. Developing TRL models in free settings is more challenging than in structured settings due to (i) irregular 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.

To understand these kinds of challenges, we present the average pair-wise distribution of trajectories in the ETH dataset [37], which captures trajectories in a free setting, annotated by group membership. Figure 5.1 presents the pairwise Euclidean distances between trajectories of the same group, i.e., positive samples (S_p), and different groups, i.e., negative samples (S_n). The red line shows where these two distances, i.e., S_p and S_n, are equal. In general, one might assume that, given individuals' tendency to walk closer to the group they belong to, all samples within the same group should have smaller S_p values compared to their S_n (i.e., all samples to be positioned above the red line). However, here we observe that a large number of samples maintain higher average S_p values than their minimum S_n values being positioned below the red line. Many representation models, particularly auto-encoders [194], which learn embeddings without explicit differentiation

between similar and dissimilar samples, may struggle to resolve this complexity. This will, in turn, limit their performance in downstream tasks.

Furthermore, recent studies in modeling movements in structured settings implement spatio-temporal pre-processing methods such as behavioral feature extraction [194], grid-cell projection [195], and graph embeddings [193]. 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 depends on the choice of hyperparameters, such as cell size. In a free setting, finding the optimal grid size is more challenging, especially in the presence of imprecision due to noise.

This paper presents a robust trajectory representation learning framework called SiamCircle, which is directly applicable 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 similarity class and enriches datasets with a limited sample size.
- 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 nine evaluation metrics in downstream task using three classes of metrics, namely trajectory ranking, trajectory similarity, and clustering. We compare the performance of SiamCircle to three other baseline models.

The remaining part of the paper is organized as follows. In Section 5.2, we discuss related work. Section 5.3 presents the problem statement, and Section 5.4 explains

the details of our proposed method. The experimental setup and results are presented in Section 5.5 and Section 5.6, respectively. Finally, we summarize the paper and discuss future research directions in Section 5.7.

5.2 Related Work

Several studies proposed TRL models primarily in structured settings like urban traffic forecasting and detecting transportation modes [196, 197]. Recent studies incorporated external data, such as road networks and urban structures, into TRL models [198, 199]. While these models show promise, they become impractical for data collected in free settings, especially in the absence of external information. Moreover, several studies adopt feature engineering techniques to transfer raw trajectories into an abstract representation before training their model [193, 194]. Trajectory2vec [194] proposed a manual feature extraction algorithm to extract space- and time-invariant characteristics of trajectories, e.g., change of rate of turns. This model adopted a sequence-to-sequence auto-encoder to generate a deep representation of trajectories from extracted features.

Moreover, TRL via Contrastive learning has been employed in various machine learning problems in natural language and image processing and, more recently, in the context of spatial problems [193, 200, 201]. Contrastive learning is especially beneficial when using noisy and imperfect datasets with limited sample sizes. Fan Z. et al. [201] proposes a contrastive learning approach powered by Triplet loss [202] in combination with a sliced encoder network for a user re-identification problem based on trajectory data. Their method, S-BiLSTM, learns the robust part of individuals' trajectories (segmented in 24 hours). Such a 24-hour sliced design is not applicable for movements in free settings where trajectories do not exhibit regular patterns. TrajCL [193] is another contrastive learning model including a dual-feature multi-head self-attention-based encoder with InfoNCE loss [203] 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 various similarity ranges an equal pre-defined margin.

The present study uses a Siamese network architecture 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 [193, 194] to limit the unnecessary loss of data and oversimplification in free settings. We further implement a dynamic penalty scheme based on the degree of similarity to obtain a

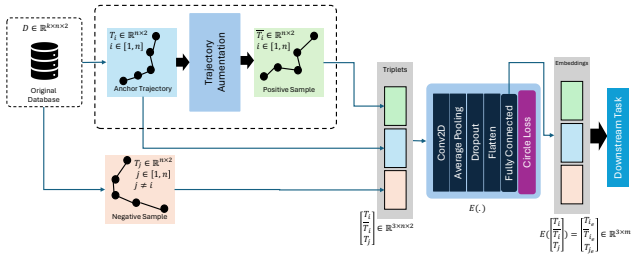

Figure 5.2: An overview of SiamCircle: Data Augmentation Process to create the [Anchor, Positive, and Negative] triplets, Siamese Network with Circle Loss to generate the embeddings of the given raw triplets.

sustainable optimization process and high-quality embeddings to address the rigidness problem mentioned above in earlier works (see, e.g., [193, 194]).

5.3 Problem Definition

Our trajectory modeling problem is based on a given trajectory dataset D, where each trajectory $\{T_i \in D | T_i = \{(x_1, y_1), \cdots, (x_n, y_n)\}\}$ is a sequence of two-dimensional points recording the trace of the moving object i with a fixed size n where (x_n, y_n) is the n^{th} location of the object.
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. Here, $f(\cdot, \cdot)$ is a distance function (e.g., Euclidean distance or other distance measures).

5.4 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 Figure 5.2.

5.4.1 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 pre-defined transformations to the original data, for instance, by truncating or point masking [193, 195]. 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 Equation 5.1:

$$\bar{T}_i(\bar{x}, \bar{y}) = (x_i + x'_i, y_i + y'_i), \qquad \begin{cases} x'_i = \mathcal{U}(l_x, h_x).d_s, \\ y'_i = \mathcal{U}(l_y, h_y).d_s, \end{cases} \tag{5.1}$$

where $\bar{T}_i(\bar{x}, \bar{y})$ is the augmented trajectory of moving object i. x_i is the x coordinate of original trajectory T_i. x'_i is the distortion vector across x coordinates. $\mathcal{U}$ is the uniform distribution bounded between lower bound $l_x = -\sigma(\Delta x_i)$ and higher bound $h_x = \sigma(\Delta x_i)$, in which $\sigma(.)$ is the standard deviation. Δx_i is the positional differences across x coordinates. d_s is a distortion scale randomly selected among the given scaling range. In all the mentioned annotations, replacing x with y gives the same definition across the y coordinate (instead of x). In our experiments, two types of d_s are used, namely *additive noise* and *jumps*. While additive noise applies to all coordination 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, $\bar{T}_i(\bar{x}, \bar{y})$, creates one variation of the object i in the database per distortion scale. In Section 5.4.2, we will discuss our proposed neural network model.

5.4.2 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 [204]. 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 neural networks in time-series forecasting [205, 206], the 2-D convolutional layer is deployed as the first layer to extract feature maps by sliding the learning filters through the input. This section is further developed by an average-pooling layer, which reduces the noise and helps the network generalize better, and a Dropout layer to prevent overfitting. Overall, this section casts the trajectory into a more compact representation while retaining the most significant features in trajectories.

Fully-connected Layer. The outputs from the previous layer representing high-level embedding features are fed to a Fully-connected layer. This allows the model to learn the non-linear combinations of these features. The generated embeddings will be given to Circle Loss to allow the training process as detailed in Section 5.4.3.

5.4.3 Loss Function: Circle Loss

Most loss functions, including the triplet loss [202], directly incorporate the pairwise distances, i.e., s_n and s_p, into a similarity score. This optimization approach is rigid as it enforces equal penalty strength on each similarity score. 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. [207] 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:

$$L = log\left[1 + \sum_{j=1}^{L} exp\left(\gamma\alpha_n^j(s_n^j - \Delta_n)\right)\sum_{i=1}^{K} exp\left(-\gamma\alpha_p^i(s_p^i - \Delta_p)\right)\right], \begin{cases}\alpha_p^i = [O_p - s_p^i]_+, \\ \alpha_n^j = [s_n^j - O_n]_+, \end{cases} \tag{5.2}$$

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, $[\mathring{u}]_+$ 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 Equation 5.2. 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 for TRL in a triplet learning paradigm using a Triplet-based 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 [202]. 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 the model.

5.5 Experiments

We evaluate SiamCircle on five real trajectory datasets using nine evaluation metrics in downstream task using three classes of metrics: trajectory ranking, trajectory similarity, and clustering. The following sections describe the experimental settings,

datasets, baselines, and evaluation metrics.

Experimental Settings. SiamCircle is implemented in Tensorflow.[1] In the data augmentation, jumps are only applied randomly to $k \in [2, 5]$ number of samples. The additive noise is $d_s \in [0.5, 1]$, and the jump scale is $d_s \in [15, 20]$. In the convolutional layer, the filter size is 64, and the kernel size is (5, 5). The average pooling size is (2,2), and the dropout rate is 0.2. The dimension of the embedding space is $d = 10$. In Circle Loss, we set γ=8 and m=0.85. We report the average and standard deviation of results across ten runs for each experiment. The Wilcoxon signed rank test [208] 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. The SiamCircle is optimized with Adam Optimizer. The learning rate is initialized to 0.001 and decayed by half after every 5 consecutive epochs with no improvement in the loss. The maximum number of training epochs is set to 1000 with an early stop after 10 consecutive epochs without improvements in the loss. In the following sections, we present the details of datasets, baselines, and evaluation metrics used in our study.

5

Datasets. Five pedestrian datasets —*eth*, *hotel* [209], and *zara01*, *zara02*, and *students03* [210]—are utilized in the experiments. These are widely recognized benchmarks for group detection tasks using spatio-temporal data [37]. They contain the location and velocity of movements across multiple timeframes, including ground truth information on group membership. All five datasets are captured in a free setting. Table 5.1 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 Section 5.4.1 to generate augmented trajectories per class (or group). The augmented data is only used during the training process. Hence, we chose a higher ratio for the test dataset to create enough samples for our evaluation experiments. Specifically, first, we split the original data set into a train and test set (i.e., train:test, 40:60). Next, via our proposed data augmentation process, we further enriched the training dataset by generating a larger sample size (over 5000 samples) with higher variability, i.e., different distortion and jumps scales. The generated augmented trajectory dataset is used only in the training process, and the remaining original dataset has been used for evaluation.

Baseline. We compared SiamCircle with three other deep learning models, namely S-BiLSTM [201], TrajCL [193], and Trajectory2vec [194]. 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 [201]. In

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

Table 5.1: 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)

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 nine evaluation metrics in downstream task using three classes of metrics. Specifically, the trajectory ranking task is evaluated using the top-k hitting ratio (k $=$ 1, 5, 10). The trajectory similarity task is assessed using the *most similar search (MSS)* and *cross-similarity (CS)*. Finally, the trajectory clustering task is evaluated using two supervised metrics, i.e., Normalized Mutual Information (NMI) [211] and Fowlkes Mallows (FM) [212], and two unsupervised metrics, i.e., Davies Bouldin (DB) [127] and Silhouette (Si) [123]. The summary of these evaluation metrics is presented 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.

Most Similar Search (MSS) [195]. This metric measures the quality of self-similarity among trajectories. Specifically, we randomly select two trajectories per class and store them in two separate subsets. Thus, each subset's size equals the number of classes (or groups) in each test dataset. Then, we calculate the pairwise distances between the embeddings of all trajectories in these two subsets and sort the obtained distances in ascending order per class. Ideally, the two trajectories from the same class should rank first among others. We calculated the average rank of the same-class trajectories across all classes and reported it as the MSS score. The lower score shows a better performance.

Cross-Similarity (CS) [195]. This metric evaluates the cross-similarity performance by measuring the extent to which a similarity measure maintains the distance between two trajectories from different classes, i.e., cross-similarity. CS score is formulated in Equation 5.3:

$$CS = \frac{|d(T_a, T_b) - d(\overline{T}_a, \overline{T}_b)|}{d(T_a, T_b)} \tag{5.3}$$

where T_a and T_b are the embeddings of two distinct original trajectories from the test dataset. $\overline{T}_a$ and $\overline{T}_b$ are the embeddings of their distorted counterparts respectively. $d(.,.)$ denotes the Euclidean distance between the two given trajectories. A smaller CS measure shows better performance.

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, namely NMI [211] and FM score [212], and two unsupervised metrics, i.e., DB score [127] and Si [123], to evaluate the performance of a K-mean Clustering Algorithm [213]. The number of ground truth groups, as indicated in Table 5.1, 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.

5.6 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 component analysis to investigate the impact of different neural network components and loss function analysis to investigate the impact of different loss functions in our framework.

Component analysis: As described before, our model mainly includes one 2-D convolutional layer to extract spatio-temporal features in both dimensions of trajectories (i.e., $1 \times Conv2D$). Through this experiment, we compare the performance of $1 \times Conv2D$ with three other alternatives. Since in the literature, recurrent

neural networks have been suggested for analyzing time-series data, we have explored the use of LSTM and GRU layers instead of convolutional layers in models *LSTM*, and *GRU*. Moreover, we explored the performance of two 2-dimensional convolutional layers (instead of one) in $2 \times Conv2D$. As shown in Table 5.2, the model $1 \times Conv2D$, which includes one layer of 2-dimensional convolutional layer, on average, ranked higher than all the other models in all classes of metrics across all datasets. This demonstrates that the $1 \times Conv2D$ component captures spatial and temporal dependencies in the data. Hence, this model is used in the following sections to conduct further experiments.

Loss function analysis. After selecting the best-performing model in terms of neural network design, in this section, we investigate the impact of different loss functions on the performance of the selected model. Thus, we trained our proposed neural network design, i.e., $1 \times Conv2D$, with Triplet loss [202], Contrastive loss [214], USR loss [215], and Circle Loss. As shown in Table 5.3, 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 5.1). This might explain the poor performance of these two metrics, as both indicate how well clusters are separated. This task imposes more challenges in small areas and makes the clusters less distinguishable. 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 three different classes of metrics, including trajectory ranking, trajectory similarity, and clustering, using the nine evaluation metrics that are explained in previous sections. The results are depicted in Table 5.4.

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 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.

Trajectory Similarity. In the Trajectory Similarity task, none of the models con-

Table 5.2: The result of the ablation study in component analysis. The highest ranks in each evaluation metric are shown in Boldface. R denotes the performance ranking. The M_R shows the average rank per category. Categories, separated by horizontal lines, include trajectory ranking, trajectory similarity, supervised clustering, and unsupervised clustering.

Loss Function	Metric	Datasets					R	M_R
		ETH	Hotel	Student03	Zara01	Zara02		
LSTM	Hit-1	0.792	0.581	0.703	0.7	0.711	2.5	2.8
	Hit-5	0.773	0.884	0.706	0.798	0.901	2.8	
	Hit-10	0.807	0.919	0.699	0.839	0.874	3.2	
	MSS	2.694	1.867	2.977	2.912	2.78	3.4	3.0
	CS	0.617	2.968	1.389	3.388	2.206	2.6	
	NMI	0.85	0.847	0.911	0.785	0.841	2.4	2.4
	FM	0.361	0.423	0.392	0.165	0.091	2.4	
	DB	0.685	0.602	0.54	0.607	0.313	**2.2**	2.7
	Si	0.322	0.254	0.32	0.238	0.368	3.2	
GRU	Hit-1	0.76	0.452	0.738	0.738	0.725	**2.2**	2.6
	Hit-5	0.812	0.89	0.726	0.775	0.889	2.5	
	Hit-10	0.848	0.932	0.743	0.811	0.866	3.0	
	MSS	2.528	1.6	2.754	3.441	2.712	2.4	2.6
	CS	0.9	2.983	1.34	3.988	1.906	2.8	
	NMI	0.828	0.83	0.902	0.79	0.836	3.2	3.2
	FM	0.307	0.37	0.355	0.179	0.074	3.2	
	DB	0.726	0.613	0.517	0.573	0.323	**2.2**	2.3
	Si	0.326	0.285	0.305	0.243	0.377	2.4	
2×Conv2D	Hit-1	0.625	0.613	0.645	0.712	0.711	3.1	2.8
	Hit-5	0.792	0.89	0.741	0.767	0.858	2.7	
	Hit-10	0.865	0.961	0.742	0.822	0.824	2.6	
	MSS	2.222	1.8	2.6	3.676	2.932	2.8	2.4
	CS	1.542	2.495	0.907	6.217	1.119	**2.0**	
	NMI	0.83	0.77	0.923	0.785	0.832	3.1	3.1
	FM	0.308	0.198	0.464	0.146	0.05	3.2	
	DB	0.708	0.6	0.552	0.625	0.325	3.2	2.7
	Si	0.325	0.218	0.322	0.254	0.393	2.3	
1×Conv2D	Hit-1	0.771	0.523	0.686	0.724	0.732	**2.2**	**1.8**
	Hit-5	0.84	0.88	0.738	0.791	0.909	**2.0**	
	Hit-10	0.892	0.953	0.744	0.843	0.877	**1.2**	
	MSS	2.211	1.56	2.687	2.806	2.717	**1.4**	**2.0**
	CS	2.229	2.864	1.039	5.172	1.502	2.6	
	NMI	0.858	0.878	0.911	0.811	0.849	**1.3**	**1.2**
	FM	0.39	0.517	0.399	0.223	0.102	**1.2**	
	DB	0.627	0.671	0.526	0.589	0.324	2.4	**2.2**
	Si	0.328	0.218	0.331	0.287	0.366	**2.1**	

Table 5.3: The result of the ablation study in loss function analysis. The highest ranks in each evaluation metric are shown in Boldface. R denotes the performance ranking. The M_R shows the average rank per category. Categories, separated by horizontal lines, include trajectory ranking, trajectory similarity, supervised clustering, and unsupervised clustering.

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	
	MSS	1.972	1.4	3.9	3.147	2.949	1.8	1.8
	CS	1.9	2.762	1.968	6.197	1.13	**1.8**	
	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	
	MSS	2.389	1.733	3.046	3.912	3.712	2.8	2.8
	CS	2.316	10.926	0.972	21.021	1.028	2.8	
	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	
	MSS	3.722	1.8	7.085	5.912	7.305	4.0	3.7
	CS	2.254	3.315	2.289	8.919	1.934	3.4	
	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**	
	MSS	2.211	1.56	2.687	2.806	2.717	**1.4**	**1.7**
	CS	2.229	2.864	1.039	5.172	1.502	2.0	
	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	

sistently outperformed in either of the metrics, i.e., MSS and CS. Yet, SiamCircle ranked highest in the MSS score, and Trajectory2vec ranked highest in the CS score. This means that SiamCircle was more successful in identifying trajectories belonging to the same class but faced difficulty distinguishing trajectories from different

Table 5.4: The result of the comparative study in the format of (mean±standard division). The statistically significant results are shown by *. The highest ranks in each evaluation metric are shown in Boldface. R denotes the performance ranking. The M_R shows the average rank per category. Categories, separated by horizontal lines, include trajectory ranking, trajectory similarity, supervised clustering, and unsupervised clustering.

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	
	MSS	2.867±2.448	1.6±0.841	3.315±6.792	3.056±2.275	3.992±4.735	2.0	2.2
	CS	4.567±0.854	$0.594 \pm 0.127^*$	$0.635 \pm 0.148^*$	24.026±14.074	1.566±0.44	2.4	
	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	
	MSS	4.153±6.816	2.87±1.56	21.125±34.181	9.103±10.186	5.36±8.493	3.8	3.4
	CS	4.966±1.819	27.072±17.447	1.976±1.789	3.236±1.477	1.356±0.243	3.0	
	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	$0.405 \pm 0.024^*$	0.335±0.108	$0.073 \pm 0.022^*$	**1.4**	**1.2**
	Si	0.435±0.056	0.412±0.082	$0.436 \pm 0.048^*$	$0.438 \pm 0.066^*$	$0.518 \pm 0.007^*$	**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	
	MSS	5.572±5.282	1.953±1.179	15.701±16.687	5.624±6.107	4.766±4.223	3.2	2.6
	CS	1.187 ± 0.018	1.174±0.023	0.973±0.016	$1.424 \pm 0.014^*$	2.203±0.219	**2.0**	
	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	$0.149 \pm 0.01^*$	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	$0.771 \pm 0.420^*$	0.523±0.449	$0.686 \pm 0.464^*$	$0.724 \pm 0.447^*$	$0.732 \pm 0.443^*$	**1.2**	**1.0**
	Hit-5	$0.840 \pm 0.144^*$	$0.880 \pm 0.118^*$	$0.738 \pm 0.195^*$	$0.791 \pm 0.146^*$	$0.909 \pm 0.116^*$	**1.0**	
	Hit-10	$0.892 \pm 0.086^*$	$0.953 \pm 0.055^*$	$0.744 \pm 0.155^*$	$0.843 \pm 0.1^*$	$0.877 \pm 0.146^*$	**1.0**	
	MSS	$2.11 \pm 2.434^*$	1.56±0.828	$2.687 \pm 3.702^*$	2.806±2.659	$2.717 \pm 2.358^*$	**1.0**	**1.8**
	CS	2.229±0.063	2.864±0.046	1.039±0.128	5.172±0.432	1.502±0.151	2.6	
	NMI	$0.849 \pm 0.009^*$	0.858±0.026	$0.921 \pm 0.004^*$	$0.805 \pm 0.007^*$	0.838±0.003	**1.8**	**1.7**
	FM	$0.366 \pm 0.025^*$	0.458±0.083	$0.451 \pm 0.019^*$	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	

classes. Trajectory2vec uses an autoencoder with no explicit labeling to reconstruct representations. As a result, trajectory representation is solely based on the individual trajectory itself, without considering similarities with trajectories from the same class. This might explain why Trajectory2vec has performed better in distinguishing dissimilar trajectories (i.e., CS score) and did not perform at the same level as SiamCircle in identifying similarities among same-class trajectories (i.e., MSS score). Overall, SiamCircle obtained the best average rank M_R in the Trajectory Similarity task compared to the baselines.

Clustering. SiamCircle performed higher than the other three baselines in supervised clustering measures, i.e., NMI and FM scores (with S-BiLSTM performing similarly to SiamCircle in the NMI metric). However, this was not the case for 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, the TrajCL has ranked best in unsupervised clustering metrics. This could be because TrajCL did not use group membership during training. 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 data 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 two classes of metrics, Trajectory Ranking and Trajectory Similarity. Meanwhile, 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.

5.7 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 nine evaluation metrics in downstream tasks using three classes of metrics, namely trajectory ranking, trajectory similarity, and clustering. 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. improvements over the existing measures. SiamCircle can be used in various tasks, such as analyzing social interactions in free settings, e.g., schoolyards and sports clubs. In future research, automating hyperparameter selection in Circle Loss based on the characteristics of the given datasets can be explored.