Contrastive Pre-training with Adversarial Perturbations for Check-in Sequence Representation Learning

Letian Gong[1, 2], Youfang Lin[1, 2], Shengnan Guo[1, 2*], Yan Lin[1, 2], Tianyi Wang[1], Erwen Zheng[1], Zeyu Zhou[1], Huaiyu Wan[1, 2]

[1]School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
[2]Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing, China
{gonglt, yflin, guoshn, ylincs, wangtianyi, zhengerwen, zeyuzhou, hywan}@bjtu.edu.cn

Abstract

A core step of mining human mobility data is to learn accurate representations for user-generated check-in sequences. The learned representations should be able to fully describe the spatial-temporal mobility patterns of users and the high-level semantics of traveling. However, existing check-in sequence representation learning is usually implicitly achieved by end-to-end models designed for specific downstream tasks, resulting in unsatisfactory generalizable abilities and poor performance. Besides, although the sequence representation learning models that follow the contrastive learning pre-training paradigm have achieved breakthroughs in many fields like NLP, they fail to simultaneously consider the unique spatial-temporal characteristics of check-in sequences and need manual adjustments on the data augmentation strategies. So, directly applying them to check-in sequences cannot yield a meaningful pretext task. To this end, in this paper we propose a contrastive pre-training model with adversarial perturbations for check-in sequence representation learning (CACSR). Firstly, we design a novel spatial-temporal augmentation block for disturbing the spatial-temporal features of check-in sequences in the latent space to relieve the stress of designing manual data augmentation strategies. Secondly, to construct an effective contrastive pretext task, we generate "hard" positive and negative pairs for the check-in sequence by adversarial training. These two designs encourage the model to capture the high-level spatial-temporal patterns and semantics of check-in sequences while ignoring the noisy and unimportant details. We demonstrate the effectiveness and versatility of CACSR on two kinds of downstream tasks using three real-world datasets. The results show that our model outperforms both the state-of-the-art pre-training methods and the end-to-end models.

Introduction

Location-based services (LBS) platforms, such as Gowalla, Foursquare, Alipay, have grown rapidly in recent years. They provide convenience for people to share and find location information as well as surrounding services, meanwhile generating a large amount of human mobility behavior data, *i.e.*, check-in sequences on point of interests (POIs). This gives us an opportunity to mine and understand human mobility behavior to support various downstream practical tasks, *e.g.*, next check-in location predictions for personalized recommendation, trajectory user link, and abnormal trajectory detection for safety control. A core step of mining human mobility data is to learn representations for check-in sequences. In order to help for improving downstream tasks, the learned representations should be able to fully describe the spatial-temporal mobility patterns of people and capture the high-level semantics of check-in sequences. Actually, representation learning is always one of the hottest topics in the fields of spatial-temporal data mining (STDM) (Lin et al. 2021) (Wan et al. 2021), natural language processing (NLP) and computer vision (CV) (Le-Khac, Healy, and Smeaton 2020). Many successful self-supervised representation learning algorithms that utilize numerous unlabelled data to capture the universal patterns of studied data are proposed and are used to improve a wide range of downstream tasks. Among these algorithms, contrastive pre-training via self-supervised signals (Jaiswal et al. 2020) is proven to be the most effective way in many scenarios. It pre-trains encoders by contrasting positive pairs and negative pairs generated from data augmentations. Compared to the generative self-supervised models for representation learning that aims to recover all the details (usually including the noise) of original, contrastive self-supervised models distill high-level semantics from sparse and noisy inputs by "learning to compare" (Liu et al. 2021) and usually enjoy satisfactory generalizable abilities with the help of pre-training paradigm. Inspired by these success, we want to explore adopting the contrastive pre-training paradigm for check-in sequence representation learning. In this way, we expect that universal spatial-temporal patterns and high-level semantics could be captured and the pre-trained representation model could improve the computational efficiency and the overall prediction performance of downstream tasks.

The key to learning representations for check-in sequences via contrastive pre-training is to design an effective data augmentation strategy so that it can encourage the encoder to mine the spatial-temporal patterns from mobility behavior data and capture the high-level semantics for human mobility. However, existing augmentation methods including those specially designed for sequential data and spatial-temporal data cannot meet the needs. Specifically, there are two reasons.

*Corresponding author: Shengnan Guo

1) Each check-in point in the sequence is simultaneously decided by the happening time and the POI. Along the spatial dimension, each POI is discrete, highly semantic, and information-dense; Along the temporal dimension, happening time is sparse and full of uncertainty. Thus, directly adopting existing data augmentation methods (Lin et al. 2021; Wan et al. 2021), *e.g.*, replacement, masking, disruption, and cropping to disturb the POIs in sequences fail to generate high-level semantically similar positive samples. For example, as shown in Figure 1(a), replacing the POI of a check-in, *i.e.*, library with a nearby KTV will change the semantics and the moving purpose of a trip. On the other hand, the widely adopted data augmentation methods for continuous data such as adding random noise cannot bring much difference to the check-in sequences, as shown in Figure 1(b). Consequently, the representation learning model cannot benefit a lot from such data augmentation strategies.

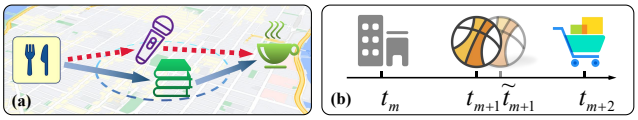


Figure 1: (a) spatial augmentation for POI by replacement. (b) temporal augmentation for time by adding noise.

2) Each check-in sequence has unique characteristics, (*e.g.*, length, POI sets), making them easily distinguishable. This will make the corresponding contrastive pretext task meaningless. Take Figure 2 for example. Two check-in sequences in the same training batch are totally different in lengths, moving areas, and visited POIs. This means that augmented samples are also easily distinguishable.

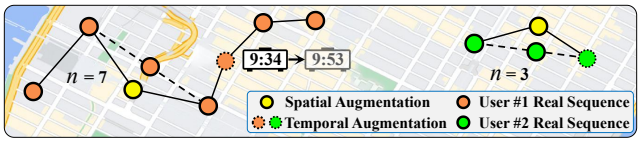


Figure 2: Most (augmented) sequences in the same batch are easily distinguishable.

In order to tackle the above challenges, we propose a novel ***C**ontrastive pre-training method using **A**dversarial perturbations for **C**heck-in **S**equence **R**epresentation Learning* (**CACSR**). In CACSR, we first propose a spatial-temporal data augmentation block for perturbing check-in points in the latent space. This block relieves the stress of designing manual data augmentation strategies for check-in data. Secondly, in order to make the contrasting process more efficient for guiding the training of the model, we develop adversarial training to generate "hard" positive and negative pairs for check-in sequences. Specifically, we generate negative examples by implementing a small perturbation while minimizing semantic-level conditional likelihood. Besides, we generate positive examples by adding large perturbations while enforcing them to have high conditional likelihood. In this way, negative samples are very close to the corresponding anchors in the latent embedding space while having a large gap in the semantics (*e.g.*, traveling purposes), and positive samples are far apart from the corresponding anchors while preserving similar semantics. As a result, the model will be encouraged to understand the high-level semantics of each check-in sequence, capture the universal spatial-temporal patterns of human mobility, and generate accurate representations for check-in sequences. Overall, the contributions of this work are summarized as follows.

- we propose a way to perform spatial-temporal data augmentation on the spatial and temporal features of check-ins in embedding spaces, which avoids tedious manual adjustment on data augmentation strategies for preserving semantics.
- To tackle the ineffectiveness of the existing augmentation approach when applied for check-in sequences, we adopt an adversarial strategy to automatically generate "hard" negative and positive pairs. This encourages the model to capture the high-level spatial-temporal semantics of mobility behavior instead of the meaningless characteristics.
- We evaluate CACSR on three real-world check-in datasets for two downstream tasks. The experimental results prove the superiority and versatility of our model.

Related Work

Mobility Data Mining The development of location-based services creates a new research topic, *i.e.*, mobility data mining, among which next location prediction (LP) and trajectory user link (TUL), are two important tasks to help improving the quality of services. Around these tasks, recent studies have proven that deep learning, especially RNNs and attention mechanisms are effective ways to capture the sequential and periodic patterns of human mobility. The most representative models for LP include DeepMove (Feng et al. 2018), STAN (Luo, Liu, and Liu 2021), LSTPM (Sun et al. 2020), and SERM (Yao et al. 2017). The most representative models for TUL are TULER (Gao et al. 2017) and DeepTUL (Miao et al. 2020). However, these end-to-end supervised methods designed for specific tasks are not universal. More importantly, since the training of these models is driven by labels, the model's ability of understanding high-level semantics in check-in sequences is not spurred.

Pre-training and Contrastive Learning The core of these mobility mining tasks is to learn the representation of check-in sequences. Many related works prove the effectiveness of applying the pre-training paradigm to perform check-in sequence representation learning. *e.g.*, TULVAE (Zhou et al. 2018) and MoveSim (Feng et al. 2020) respectively employ Variational Auto-Encoder and Generative Adversarial Network to capture the underlying patterns of check-in sequences via the pre-training. Recently, pre-training through contrastive learning is proven to be effective in sequential data modeling. Contrastive learning enables the model to capture high-level semantics meanwhile ignoring the noisy details of sequences. It trains models in a self-supervised manner by comparing positive and negative pairs from data augmentation. In NLP, representative contrastive pre-training models differ in data augmentation strategies. *e.g.*,

SimCSE (Gao, Yao, and Chen 2021) augments data with the help of dropout operations; ConSERT (Yan et al. 2021) augments data by disrupting, slicing, and deleting the representation in the hidden space; VaSCL (Zhang et al. 2022) enhances the discriminative power of the model by introducing difficult negative samples; CLAPS (Lee, Lee, and Hwang 2020) introduces adversarial perturbations to generate indistinguishable augmented samples, and finally largely enhances its robustness and discrimination ability. In the field of mobility mining, SML (Zhou et al. 2021) is the first to adopt contrastive learning. It directly applies the widely used data augmentation strategies, such as cropping or replacement, on check-in sequences while ignoring the unique characteristics of check-in sequences, thus cannot perform well enough.

Motivated by the studies mentioned above, considering the characteristics of check-in sequences, we will investigate how to design a robust and universal contrastive pre-training model for check-in sequence representation learning with the help of adversarial perturbations.

Preliminaries

Definition. *Check-in Sequence.* Check-in sequences describe the moving of users among POIs during a certain period. A check-in sequence is denoted as $\mathcal{T} = (r_1, r_2, \cdots, r_n)$, where $r = (t, p)$ represents a check-in happening at $p \in \mathcal{P}$ at timestamp t. $\mathcal{P}$ denotes the POI set, where each POI p refers to a geographical location with unique latitude, longitude, and semantic functions.

Problem Statement. *Pre-training Representation for check-in Sequence.* The goal of this paper is to pre-train a parameterized encoder G to generate a contextual representation for a given check-in sequence $\mathcal{T}$, *i.e.*, $G(\mathcal{T})$. Specifically, the encoder G is first trained within an adversarial framework in a self-supervised manner with no task-specific objectives and then is applied to various downstream tasks, such as next Location Prediction (LP), Trajectory User Link (TUL), *etc*, so that the downstream tasks could benefit from the universal encoder.

The CACSR Model

Figure 3 illustrates the architecture of our proposed CACSR model. Firstly, our CACSR encodes check-in sequences of POIs into latent representations by mining and exploiting the spatial-temporal features of check-ins. Then, following the contrastive learning framework (Lee, Lee, and Hwang 2020), our model generates positive and negative pairs, and is finally trained by maximizing the similarity between positive pairs, while minimizing the similarity between the negative pairs.

Our contributions lie in how to generate effective positive and negative pairs by considering the unique characteristics of check-in sequences, through which we could build a meaningful contrastive learning based pretext task. Firstly, facing the dilemma of proposing a proper augmentation strategy along the spatial and temporal dimensions in the original input space, we propose to make spatial-temporal augmentation in the latent space to preserve the macro spatial-temporal patterns and the high-level semantics of check-in sequences. Secondly, considering that each check-in sequence has unique characteristics, (*e.g.*, length, POI sets), making them easily distinguishable and contrastive training ineffective,

we propose to adopt adversarial perturbation to generate difficult negative and positive pairs. Trained with these "hard" negative and positive pairs, our model could effectively capture the semantics of check-in sequences and enjoy satisfactory generalization ability and robustness.

Next, we will detail the basic encoder for capturing the spatial-temporal correlations between check-ins, the data augmentation algorithms (*i.e.*, positive and negative pairs generations), and the design of the contrastive training loss.

Encoder Module

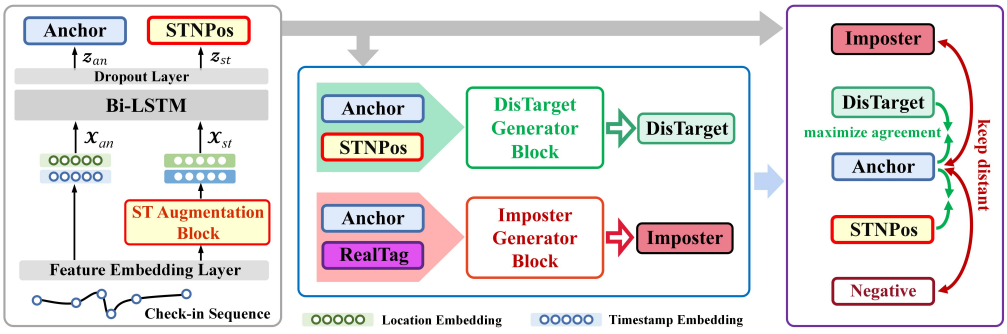
In this section, we introduce how to encode the original check-in sequences. As shown in Figure 3(a), a check-in sequence $\mathcal{T} = (r_1, r_2, \cdots, r_n)$ is encoded into its latent representation $\boldsymbol{z}$ through feature embedding layers and Bidirectional Long Short-Term Memory layers (BiLSTMs).

The feature embedding layers transform the original spatial-temporal features into dense representations. Specifically, the original spatial feature of a check-in, *i.e.*, POI p is represented by a $|\mathcal{P}|$-dimensional one-hot vector. The original temporal feature, *i.e.*, happening timestamp t is first discretized with an hour and then be represented by a T-dimensional one-hot vector (T=24). Then, we learn embeddings for each POI, denoted by $E_p \in \mathbb{R}^{|\mathcal{P}| \times d_p}$, to capture the static characteristics of each POI, and embeddings for each time slots, denoted by $E_t \in \mathbb{R}^{T \times d_t}$, to learn the semantics of each time slot via an end-to-end manner. Overall, we define the above embedding process as $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = f(\mathcal{T}; \boldsymbol{\theta}_{st})$, where $\boldsymbol{\theta}_{st}$ refers to all the learnable parameters in the embedding layer.

Then we input the embedded sequence $\mathcal{X}$ into a stack of Bidirectional Long Short-Term Memory layers (BiLSTMs), which models contextural information by leveraging two independent hidden layers along opposite directions. Finally, we use the latest hidden state $\boldsymbol{h}_n \in \mathbb{R}^{2d}$, which is the combination of the forward hidden state $\overrightarrow{\boldsymbol{h}}_n \in \mathbb{R}^d$ and the backward hidden state $\overleftarrow{\boldsymbol{h}}_n \in \mathbb{R}^d$, as the sequence representation $\boldsymbol{z}$ to summarize the mobility behavior over the sequence. In addition, to ensure the generalization ability of the model, dropout layers are further adopted.

ST Adversarial Perturbations Module

To preserve the macro spatial-temporal patterns of augmentation and let the encoder effectively capture the high-level semantics of check-in sequences, we design two kinds of data augmentation blocks. Firstly, facing the dilemma in manual adjustment on data augmentation strategies respectively for the temporal and spatial features of check-ins, we introduce how to apply disturbances to spatial-temporal features in the latent space. Then, in order to improve the difficulties of the comparisons between samples, we introduce how to dynamically generate "hard" positive and negative samples leveraging adversarial perturbation.



(a) Encoder Module (b) Adversarial Perturbation Module (c) Contrastive Module

Figure 3: The model architecture of CACSR. (a) shows the process of converting the check-in sequence into a latent representation. (b) mainly uses STNPos which are generated by ST augmentation block and anchor sequence to generate "hard" positive-negative samples by adversarial perturbations. (c) introduces the contrastive loss relationship among different samples.

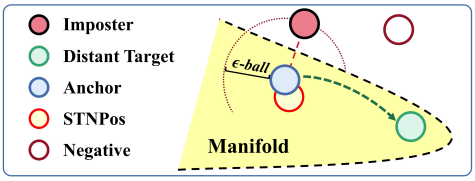


Figure 4: Data Augmentation Schematic. The yellow area represents the semantic manifold space of the anchor, and only the samples within the yellow area have the similar semantics to the anchor.

ST Augmentation Block Spatial-Temporal Augmentation block is designed to generate the corresponding positive example for a check-in sequence in the latent embedding space. We use $\mathcal{X}_{an}$ to denote the output of the feature embedding layers with $\mathcal{T}$ as input. $\mathcal{X}_{an}$ is the anchor vector in our contrastive learning algorithm. Besides, we use $\mathcal{X}_{st}$ to denote the corresponding positive example, which is generated by disturbing the parameters of the feature embedding layers. Formally,

$$\mathcal{X}_{an} = f(\mathcal{T}; \boldsymbol{\theta}_{st}), \quad \mathcal{X}_{st} = f\left(\mathcal{T}; \boldsymbol{\theta}'_{st}\right), \tag{1}$$

in which $\boldsymbol{\theta}_{st}$ and $\boldsymbol{\theta}'_{st}$ represent the parameters of feature embedding layers and the ST augmentation block, respectively. And the $\boldsymbol{\theta}'_{st}$ is defined as:

$$\boldsymbol{\theta}'_{st} = \boldsymbol{\theta}_{st} + \eta \Delta \boldsymbol{\theta}_{st}, \quad \Delta \boldsymbol{\theta}_{st} \sim \mathcal{N}\left(0, \boldsymbol{\sigma}^2\right), \tag{2}$$

where we achieve $\boldsymbol{\theta}'_{st}$ by adding a perturbation term $\Delta \boldsymbol{\theta}_{st}$ sampled from a Gaussian distribution with zero mean and variance $\boldsymbol{\sigma}^2$. η is the scale factor for the magnitude of the perturbation. $\boldsymbol{\sigma}$ and η are two hyperparameters. Then after BiLSTMs, we use $\boldsymbol{z}_{an}$ and $\boldsymbol{z}_{st}$ to denote the sequence representation for the anchor and the *spatial temporal noise positive* (**STNPos**) samples respectively.

Imposter Generator Imposter Generator is designed to generate a "hard" negative sample $\boldsymbol{z}_{im}$ called **Imposter** for the anchor $\boldsymbol{z}_{an}$. $\boldsymbol{z}_{im}$ is expected to have different semantics but similar representations to $\boldsymbol{z}_{an}$. Figure 4 illustrates the relations between them. We use the next location y (termed as **RealTag**) to describe the semantics of $\boldsymbol{z}_{an}$, since the next location can reflect the traveling purpose of the check-in sequences. Notice that y does not need introduce any extra manual labeling because "the next location" is part of the information from the check-in sequences itself. Specifically, so as to remain similar representation vectors, we generate $\boldsymbol{z}_{im}$ by adding a tiny perturbation $\boldsymbol{\delta}$ to the anchor $\boldsymbol{z}_{an}$ and limit its Euclidean norm within ϵ; Meanwhile, so as to change the original semantics as much as possible, we minimize the conditional likelihood of $\boldsymbol{z}_{im}$ with regard to y. Formally,

$$\boldsymbol{z}_{im} = \boldsymbol{z}_{an} + \boldsymbol{\delta}, \tag{3}$$

$$\boldsymbol{\delta} = \underset{\boldsymbol{\delta}, \|\boldsymbol{\delta}\|_2 \leq \epsilon}{\arg\min} \log p_{\boldsymbol{\theta}}\left(y \mid \mathcal{X}_{an}; \boldsymbol{z}_{an} + \boldsymbol{\delta}\right), \tag{4}$$

where $\boldsymbol{\delta} \in \mathbb{R}^{2d}$ has the same size with $\boldsymbol{z}_{an}$.

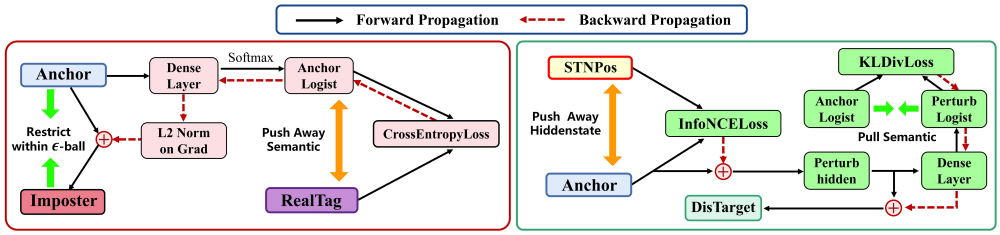
Because the exact minimization of the conditional likelihood with regard to $\boldsymbol{\delta}$ is intractable for neural networks, we implement it as follows:

$$\boldsymbol{z}_{im} = \boldsymbol{z}_{an} - \epsilon \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}, \tag{5}$$

$$\boldsymbol{g} = \nabla_{\boldsymbol{z}_{an}} \log p_{\boldsymbol{\theta}}(y \mid \mathcal{X}_{an}). \tag{6}$$

Figure 5(a) describes the detailed implementation of the imposter generator. Overall, $\boldsymbol{z}_{im}$ is an adversarial sample, since we train the model to push the semantics of $\boldsymbol{z}_{im}$ far away from that of $\boldsymbol{z}_{an}$ while constraining $\boldsymbol{z}_{im}$ within the ϵ-ball around $\boldsymbol{z}_{an}$. The generation of imposter yields a hard task for the encoder to distinguish the negative sample $\boldsymbol{z}_{im}$ from positive samples, thus it finally enhances the generalization ability and robustness of the encoder.

Distant Target Generator Distant Target Generator is designed to generate an extra hard positive example $\boldsymbol{z}_{dis}$ called *distant target* (**DisTarget**) for the anchor $\boldsymbol{z}_{an}$. $\boldsymbol{z}_{dis}$ is expected to have similar semantics with $\boldsymbol{z}_{an}$, but they should be far away in the latent space. Figure 4 illustrates the relations between them. Specifically, we add a significant perturbation $\boldsymbol{\xi} \in \mathbb{R}^{2d}$ to $\boldsymbol{z}_{an}$ to minimize its cosine similarity with



(a) Imposter Generator Block (b) DisTarget Generator Block

Figure 5: Workflow Diagram for the Adversarial Perturbation. (a) and (b) show the forward propagation and back propagation for generating Imposter and DisTarget. And we use dense layer and softmax to calculate the probability to obtain its logist.

$\boldsymbol{z}_{an}$, while the conditional likelihood of $\boldsymbol{z}_{dis}$ with regard to y is mandated to stay at a high level. However, the accurate calculation of $\boldsymbol{\xi}$ under such restrictions is intractable likewise. Thus, we use the following two computation phases to approximate it.

As shown in Figure 5(b), we first add perturbation to $\boldsymbol{z}_{an}$ such that it minimizes the contrastive learning loss $\mathcal{L}_{\text{cont}}(\boldsymbol{\theta})$ (refer to Eq.10, in which $\boldsymbol{z}_{an}$ can be regarded as $\boldsymbol{z}_x$ and $\boldsymbol{z}_{st}$ can be regarded as $\bar{\boldsymbol{z}}_x$). Formally,

$$\tilde{\boldsymbol{z}} = \boldsymbol{z}_{an} - \eta \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \quad \boldsymbol{v} = \nabla_{\boldsymbol{z}_{an}} \mathcal{L}_{\text{cont}}(\boldsymbol{\theta}). \tag{7}$$

Afterwards, in the second stage, so as to preserve the semantics, we further pull the conditional likelihood $p_{\boldsymbol{\theta}}(y \mid \tilde{\boldsymbol{z}})$ to $p_{\boldsymbol{\theta}}(y \mid \boldsymbol{z}_{an})$ via minimizing the KL divergence between them. Formally,

$$\mathcal{L}_{KL}(\boldsymbol{\theta}) = \sum_{i=1}^{N} D_{KL}\left(p_{\boldsymbol{\theta}^*}(y \mid \boldsymbol{z}_{an}) \| p_{\boldsymbol{\theta}}(y \mid \tilde{\boldsymbol{z}})\right), \tag{8}$$

$$\boldsymbol{z}_{dis} = \tilde{\boldsymbol{z}} - \eta \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|_2}, \quad \boldsymbol{\omega} = \nabla_{\tilde{\boldsymbol{z}}} \mathcal{L}_{KL}(\boldsymbol{\theta}), \tag{9}$$

where $\boldsymbol{\theta}^*$ denotes the duplication of the encoder parameters $\boldsymbol{\theta}$, which should be detached from the computation graph so that it cannot be optimized by back-propagation. Finally, we get $\boldsymbol{z}_{dis}$ to be an additional hard positive instance for $\boldsymbol{z}_{an}$.

Contrastive Module

In accordance with the contrastive learning framework, we maximize the similarity between the anchor and the augmented positive examples, while minimizing the similarity between the anchor and the *negative examples* (termed as **Negative**) by InfoNCE Loss (Oord, Li, and Vinyals 2018),

$$\mathcal{L}_{\text{cont}}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \frac{\varphi(\boldsymbol{z}_x, \bar{\boldsymbol{z}}_x)}{\sum_{\boldsymbol{z}_{an}, \boldsymbol{z}'_x \in S} \varphi(\boldsymbol{z}_x, \boldsymbol{z}'_x)}, \tag{10}$$

where $\varphi(i, j) = \exp\left(\text{sim}(i, j)/\tau\right)$ measures the correlation between two representations, where $\text{sim}(\cdot, \cdot)$ denotes the cosine similarity function, $\bar{\boldsymbol{z}}_x$ represents the augmented positive sample. S is a set of negative samples selected randomly from the same batch.

As discussed in the introduction, training the encoder via the vanilla contrastive learning framework, *i.e.*, leveraging random non-target check-in sequences as negative examples is meaningless since many of these vanilla negative examples are situated far away from the positive examples in the embedding space. In order to make the contrasting process more efficient for guiding the training of the model, we additionally use the generated "hard" negative and positive examples to guide the model's training. Details are as follows.

Imposter Contrastive Loss We design the imposter contrastive loss by introducing the imposter representation $\boldsymbol{z}_{im}$ as an extra negative instance to Eq. 10. Formally,

$$\mathcal{L}_{\text{cont}-\text{neg}}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \frac{\varphi(\boldsymbol{z}_{an}, \boldsymbol{z}_{st})}{\sum_{\boldsymbol{z}'_x \in S'} \varphi(\boldsymbol{z}_{an}, \boldsymbol{z}'_x)}, \tag{11}$$

where $\boldsymbol{z}_{st}$ is the positive instance got through the spatial-temporal augmentation block. S' denotes $S \cup \{\boldsymbol{z}_{\text{im}}\}$.

Distant Target Contrastive Loss We further enhance the training process by adding the hard positive sample $\boldsymbol{z}_{dis}$ as follow,

$$\mathcal{L}_{\text{cont}-\text{pos}}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \frac{\varphi(\boldsymbol{z}_{an}, \boldsymbol{z}_{dis})}{\sum_{\boldsymbol{z}'_x \in S'} \varphi(\boldsymbol{z}_{an}, \boldsymbol{z}'_x)}, \tag{12}$$

Finally, the total contrastive loss is expressed as:

$$\mathcal{L}_{\text{cont}-\text{tol}}(\boldsymbol{\theta}) = \alpha \mathcal{L}_{\text{cont}-\text{neg}}(\boldsymbol{\theta}) + \beta \mathcal{L}_{\text{cont}-\text{pos}}(\boldsymbol{\theta}). \tag{13}$$

Experiments

In order to evaluate the quality of representation vectors of check-in sequences generated by our model. We incorporate these vectors into two types of downstream tasks and compare the results with other methods. The code has been released at: https://github.com/LetianGong/CACSR

Dataset

To demonstrate the superiority of our proposed model, our experiments are carried out on three real-world datasets derived from the raw Gowalla, and Foursquare of New York City (NYC) and Jakarta (JKT) check-in data.

In order to facilitate the training of our model, we screen out relatively high-quality trajectory sequences through some conditions. For all datasets, we take the historical time as long as no more than 120 days. We filter out users with at least 10 records, and places visited at least 10 times. We split all datasets at ratio 6 : 2 : 2 into training sets, validation sets, and test sets by the samples. Table 1 shows the statistics of three datasets after the pre-processing.

	Gowalla	NYC	JKT
#users	5,016	1,596	3,943
#locations	4,873	5,529	9,563
#check-ins	154,253	39,975	119,317

Table 1: Statistics of datasets

Fine-tuning for Downstream Applications

We employ the training set to pre-train the CACSR and use a projection head to fine-tune on both next location prediction (**LP**) and trajectory user link (**TUL**) tasks, respectively. Then we consider these downstream tasks as multi-classification problems, as formulated below. Given a check-in sequence $\mathcal{T}^u$ from a specific user $u \in |\mathcal{U}|$, we feed it to the pre-trained encoder to obtain the check-in sequence representation $G(\mathcal{T}^u)$. Then we use a projection head $f_{\boldsymbol{\theta}}$ to predict the classification y such as the next location where user will soon arrive or the user who generated this check-in sequence, *i.e.*, $f_{\boldsymbol{\theta}}\left(G(\mathcal{T}^u), \boldsymbol{\theta}\right) \mapsto y$. We maximize the conditional log likelihood $\log f_{\boldsymbol{\theta}}(y \mid G(\mathcal{T}^u))$ for a given N observations $\{(G(\mathcal{T}^u), y)\}_{i=1}^{N}$ as follows:

$$\begin{aligned} \mathcal{L}_{MLE}(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \log f_{\boldsymbol{\theta}}\left(y \mid G(\mathcal{T}^u)\right), \\ f_{\boldsymbol{\theta}}\left(y \mid G(\mathcal{T}^u)\right) &= \operatorname{softmax}\left(\boldsymbol{W} G(\mathcal{T}^u) + \boldsymbol{b}\right). \end{aligned} \tag{14}$$

In addition, we combine the location embedding with the timestamp embedding as the global embedding during the pre-training stage. In the LP task, we combine the user embedding with the global embedding for fine-tuning in the downstream stage.

Location Prediction Methods

We cover two classic check-in prediction models and two state-of-the-art LP models to demonstrate the superiority of our model.

- **LSTM** (Hochreiter and Schmidhuber 1997) use LSTM as an encoder to infer where the user will arrive next.
- **SERM** (Yao et al. 2017) simultaneously considerably enriches the semantics to successfully capture the spatial-temporal transformation rule.
- **DeepMove** (Feng et al. 2018) is a classical check-in sequence position prediction model to capture the periodicity of trajectory motion.
- **LSTPM** (Sun et al. 2020) simulates the long-term historical with the attention mechanism and a geo-dilated RNN to enhance the short-term prediction capability.

Trajectory User Link Methods

We select two end-to-end and two pre-trained TUL task models for comparison.

- **TULER** (Gao et al. 2017) is the first study to propose the trajectory user link and simulate it using RNN.
- **TULVAE** (Zhou et al. 2018) follows the work of TULER and pre-trains the encoder by adding VAE on the prior basis.
- **MoveSim** (Feng et al. 2020) captures the temporal changes in human motion employing a generative adversarial network for trajectory pre-training.
- **DeepTUL** (Miao et al. 2020) integrates numerous aspects of user mobility to simulate complicated mobility patterns.

Baseline Representation Methods

We select two state-of-art contrastive sentence representation models using contrastive learning in the NLP field. We apply them to learn the representation of check-in sequence and serve downstream tasks.

- **SimCSE** (Gao, Yao, and Chen 2021) using a very simple dropout operation to generate positive samples for contrastive learning, and effectively enhance the model's ability to represent sentences.
- **VaSCL** (Zhang et al. 2022) is suggested to add hard negative samples into the NLP field. It uses smaller batches to get better performance.

Settings

We pre-train the encoder parameters on the training set and determine the best hyperparameter combination base on the results of downstream tasks on the validation set. The downstream task models are trained using cross-entropy loss and evaluated using Acc@k and Mean reciprocal rank (MRR). We employ PyTorch to implement the baseline models and our CACSR model. As for the parameter settings, we set all embedding sizes of all models to 256. The number layer of Bi-LSTM in CACSR model is set to 3, the hidden state size is set to 512, $\boldsymbol{\sigma} = 0.1$, the scale factor $\eta = 1$, $\epsilon = 1$, $\alpha = 0.8$, $\beta = 0.5$, and $\tau = 4$. The CACSR is pre-trained for 100 epochs on the training sets with the early-stopping mechanism of 5 patience. All trials have been conducted on Intel Xeon E5-2620 CPUs and NVIDIA RTX A5000 GPUs.

Experimental Results

Table 2 shows the comparison results of our model with other baseline models on two downstream tasks. In the LP task, we evaluate the accuracy of the next location prediction and the accuracy of the user-sequence association in TUL task. The best is shown in **bold**, and the second-best is shown as underlined.

First, our CACSR-LP and CACSR-TUL consistently and significantly outperform the current end-to-end and pre-trained models in LP and TUL tasks among all datasets. In particular, our approach improves Acc@5, Acc@20, and mean reciprocal rank (MRR) metrics in the LP task by 4.3%, 3.1%, 3.7%, and in the TUL task by 6.3%, 5.1%, 5.7% on average over the three datasets compared to the best baseline approach. These results demonstrate that our model performs very well on these downstream tasks. The strength of our model benefits from our contrastive adversarial framework that is able to encourage the encoder to mine the spatial-temporal patterns from mobility behavior data and capture the high-level semantics for human mobility. And

Datasets		Gowalla			FourSquare-NYC			FourSquare-JKT		
Metric		Acc@5 (%)	Acc@20 (%)	MRR (%)	Acc@5 (%)	Acc@20 (%)	MRR (%)	Acc@5 (%)	Acc@20 (%)	MRR (%)
Task	Method									
LP	LSTM	25.21±0.36	35.5±0.62	17.94±0.32	26.53±0.69	39.54±0.71	18.68±0.46	34.79±0.72	46.02±0.31	26.54±0.36
	SERM	29.32±0.54	45.25±0.34	20.51±0.44	28.49±0.37	43.11±0.43	20.54±0.56	37.68±0.32	54.64±0.33	29.42±0.64
	DeepMove	30.88±0.52	46.89±0.68	23.06±0.66	31.84±0.68	48.29±0.68	22.29±0.33	41.35±0.45	56.55±0.39	31.96±0.49
	LSTPM	30.04±0.47	45.22±0.48	22.80±0.66	32.80±0.59	50.27±0.32	**23.87±0.49**	43.53±0.58	57.51±0.49	32.39±0.41
	SimCSE-LP	26.69±0.67	42.54±0.57	20.95±0.69	30.54±0.68	46.32±0.51	20.66±0.61	39.92±0.68	53.26±0.37	30.54±0.32
	VaSCL-LP	28.95±0.38	44.52±0.57	22.93±0.65	32.68±0.45	48.99±0.63	22.83±0.39	41.03±0.63	54.27±0.62	31.36±0.54
	CACSR-LP	**33.32±0.47**	**47.78±0.50**	**24.50±0.59**	**33.46±0.69**	**50.35±0.62**	23.67±0.62	**44.72±0.61**	**58.47±0.42**	**33.88±0.63**
TUL	TULER	54.04±0.31	62.07±0.53	48.29±0.68	51.34±0.37	60.92±0.65	45.59±0.35	61.26±0.34	70.54±0.38	57.07±0.31
	TULVAE	55.65±0.64	63.92±0.50	52.73±0.35	53.02±0.31	61.08±0.68	46.12±0.63	65.42±0.57	73.12±0.64	61.50±0.66
	MoveSim	50.16±0.50	58.83±0.51	43.17±0.67	39.07±0.42	43.99±0.35	36.31±0.62	51.72±0.57	61.66±0.54	45.38±0.63
	DeepTUL	55.15±0.52	62.44±0.63	49.93±0.64	43.32±0.55	52.39±0.74	39.24±0.39	60.63±0.31	68.19±0.55	55.67±0.67
	SimCSE-TUL	53.02±0.31	61.08±0.32	48.12±0.37	50.29±0.34	60.05±0.47	45.23±0.36	59.07±0.72	66.01±0.63	54.43±0.32
	VaSCL-TUL	54.44±0.49	63.85±0.61	53.03±0.44	51.11±0.62	59.66±0.63	44.12±0.52	66.79±0.57	73.28±0.38	62.53±0.51
	CACSR-TUL	**59.82±0.51**	**66.46±0.72**	**55.08±0.33**	**54.62±0.54**	**63.03±0.39**	**48.32±0.49**	**69.32±0.39**	**76.47±0.58**	**65.03±0.68**

Table 2: Next location prediction (LP) and trajectory user link (TUL) performance comparison between different approaches.

the self-supervised signals can effectively mitigate the sparsity of check-in data and can spur the encoder to learn accurate representations for check-in sequences. Overall, the superiority and versatility of our method show that the adversarial contrastive framework proposed in this work is well suited for modeling check-in sequences.

Second, RNN end-to-end based approaches (*e.g.*, LSTM, SERM, TULER, *etc*.) can take advantage of recurrent neural networks to capture human movement patterns and predict the next location. From the experimental results, it can be seen that the performance of these methods is not satisfactory. DeepMove and LSTPM enhance the performance of recurrent neural networks by introducing the attention mechanism and to some extent mitigating chance bias caused by the sparsity of user mobile data. MoveSim trains the encoder by a generative adversarial network, but the boost is extremely restricted for check-in data. While the additional benefit of incorporating VAE into TULVAE for pre-training is evident. The performance of SimCSE and VaSCL on the check-in dataset conclusively proves that data augmentation techniques in NLP are not directly transferable to human movement trajectory data.

Ablation Study

To better verify the role of each component, we designed four variants of CACSR.

- **Basic:** We use the encoder to directly generate two representations of the sequence without spatial-temporal augmentation. Pre-training according to the traditional contrastive learning method.
- **+ST Augmentation:** Base on the basic model, we added the spatial-temporal Augmentation block, using $\boldsymbol{z}_{an}$ and $\boldsymbol{z}_{st}$ to pre-train the encoder.
- **w/o Imposter:** We remove the $\boldsymbol{z}_{im}$ from CACSR. The rest of the settings are the same as CACSR.
- **w/o DisTarget:** We remove the $\boldsymbol{z}_{dis}$ from CACSR. The rest of the settings are the same as CACSR.

We compare these four variants with the CACSR model on the LP and TUL downstream model. Figure 6 shows the

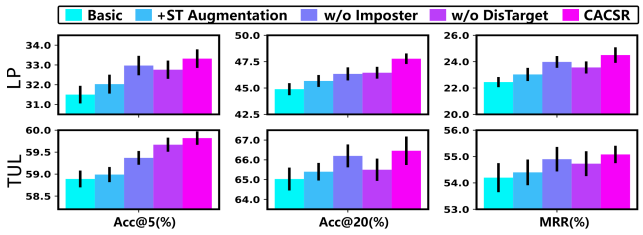


Figure 6: Component analysis of CACSR

results. In LP downstream tasks, our basic model already has high advantages. It shows that our basic model has a strong advantage in capturing the check-in sequence itself. The robustness of the model may be successfully increased by including suitable spatial-temporal augmentation perturbations in the latent space.

Clearly, it boosts the model's discriminative performance by dynamically adding impostors and distant targets. With the update of training, the difficulty of the augmented sample continues to ramp up, and the discriminative ability of the model continues to be stronger. Finally, these modules are combined in our final model to produce the best outcomes.

Conclusion

In this paper, we propose a novel adversarial contrastive model CACSR for extracting the representations of the check-in sequences. Specifically, we first propose a way to perform spatial-temporal data augmentation on the spatial and temporal features of check-ins in the latent space. Then we propose an adversarial strategy to automatically generate "hard" negative and positive pairs. These encourage the model to capture the high-level spatial-temporal semantics of mobility behavior instead of the meaningless characteristics. Experiments are conducted on the next location prediction and trajectory user link tasks with three real-world mobile user check-in datasets. The experimental results demonstrate the superiority and versatility of our model.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62202043) and China Postdoctoral Science Foundation (Grant No. 2021M700365).

References

Feng, J.; Li, Y.; Zhang, C.; Sun, F.; Meng, F.; Guo, A.; and Jin, D. 2018. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proceedings of the 2018 world wide web conference*, 1459–1468.

Feng, J.; Yang, Z.; Xu, F.; Yu, H.; Wang, M.; and Li, Y. 2020. Learning to simulate human mobility. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 3426–3433.

Gao, Q.; Zhou, F.; Zhang, K.; Trajcevski, G.; Luo, X.; and Zhang, F. 2017. Identifying Human Mobility via Trajectory Embeddings. In *IJCAI*, volume 17, 1689–1695.

Gao, T.; Yao, X.; and Chen, D. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *EMNLP (1)*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Jaiswal, A.; Babu, A. R.; Zadeh, M. Z.; Banerjee, D.; and Makedon, F. 2020. A survey on contrastive self-supervised learning. *Technologies*, 9(1): 2.

Le-Khac, P. H.; Healy, G.; and Smeaton, A. F. 2020. Contrastive representation learning: A framework and review. *IEEE Access*, 8: 193907–193934.

Lee, S.; Lee, D. B.; and Hwang, S. J. 2020. Contrastive learning with adversarial perturbations for conditional text generation. *arXiv preprint arXiv:2012.07280*.

Lin, Y.; Wan, H.; Guo, S.; and Lin, Y. 2021. Pre-training context and time aware location embeddings from spatial-temporal trajectories for user next location prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4241–4248.

Liu, X.; Zhang, F.; Hou, Z.; Mian, L.; Wang, Z.; Zhang, J.; and Tang, J. 2021. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*.

Luo, Y.; Liu, Q.; and Liu, Z. 2021. Stan: Spatio-temporal attention network for next location recommendation. In *Proceedings of the Web Conference 2021*, 2177–2185.

Miao, C.; Wang, J.; Yu, H.; Zhang, W.; and Qi, Y. 2020. Trajectory-user linking with attentive recurrent network. In *Proceedings of the 19th international conference on autonomous agents and multiagent systems*, 878–886.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Sun, K.; Qian, T.; Chen, T.; Liang, Y.; Nguyen, Q. V. H.; and Yin, H. 2020. Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 214–221.

Wan, H.; Lin, Y.; Guo, S.; and Lin, Y. 2021. Pre-training time-aware location embeddings from spatial-temporal trajectories. *IEEE Transactions on Knowledge and Data Engineering*.

Yan, Y.; Li, R.; Wang, S.; Zhang, F.; Wu, W.; and Xu, W. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. *arXiv preprint arXiv:2105.11741*.

Yao, D.; Zhang, C.; Huang, J.; and Bi, J. 2017. Serm: A recurrent model for next location prediction in semantic trajectories. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2411–2414.

Zhang, D.; Xiao, W.; Zhu, H.; Ma, X.; and Arnold, A. 2022. Virtual Augmentation Supported Contrastive Learning of Sentence Representations. In *Findings of the Association for Computational Linguistics: ACL 2022*, 864–876.

Zhou, F.; Dai, Y.; Gao, Q.; Wang, P.; and Zhong, T. 2021. Self-supervised human mobility learning for next location prediction and trajectory classification. *Knowledge-Based Systems*, 228: 107214.

Zhou, F.; Gao, Q.; Trajcevski, G.; Zhang, K.; Zhong, T.; and Zhang, F. 2018. Trajectory-User Linking via Variational AutoEncoder. In *IJCAI*, 3212–3218.