博士学位论文

时空轨迹数据的自监督学习方法研究

Self-supervised Learning of Spatial-temporal Trajectory Data

作者：林彦

导师：万怀宇

北京交通大学

2024 年 3 月

学校代码: 10004 密级: 公开

北 京 交 通 大 学

博 士 学 位 论 文

时空轨迹数据的自监督学习方法研究

Self-supervised Learning of Spatial-temporal Trajectory Data

作者姓名: 林彦 学 号: 19112020

导师姓名: 万怀宇 职 称: 教授

学位类别: 工学 学位级别: 博士

学科专业: 计算机科学与技术 研究方向: 数据与知识工程

北京交通大学

2024 年 3 月

摘要

时空轨迹通常表示为带时间戳的地点序列，是用户或车辆在地理空间区域内出行或移动行为的记录。时空轨迹数据挖掘在智能交通系统的各种应用中至关重要，这些应用包括地点分类、移动预测、交通流预测、交通需求估计、旅行时间估计、交通异常检测和轨迹-用户关联分析等。时空轨迹数据挖掘近年来成为学术研究的热点，主要受两方面因素的推动：一方面，智能手机和车辆定位系统等轨迹数据采集设备的普及，提升了大规模时空轨迹数据的可用性；另一方面，智能交通和智慧城市应用需求的增长，推动了高效时空轨迹数据挖掘方法的研发需求。

传统的机器学习算法依靠人工特征工程进行数据模式挖掘，其成本高昂，且难以有效提取时空轨迹中的复杂特征。近年来，众多研究证明神经网络与深度学习方法能够自动、有效地从数据中抽取特征。现有的时空轨迹深度学习方法大多采用端到端学习方式。然而，此方案依赖于通常难以获取的大规模标签数据集来取得最佳性能，在实际应用场景中的可行性受限。相比之下，自监督学习方法从可用性通常较高的无标签数据集中提取监督信息，学习得到的模型具有较好的泛化性，可以适配多种下游任务。在时空轨迹数据挖掘中应用自监督学习技术能够消除对人工特征工程和大规模标签数据的依赖，并且提升时空轨迹数据应用于多种下游任务时的计算与存储效率。因此，时空轨迹自监督学习逐渐成为促进轨迹数据挖掘技术与应用发展的关键研究领域。然而，时空轨迹的复杂性与特殊性给自监督学习方法的构建带来了多方面的挑战。具体而言，自监督学习方法需要对多方面时空信息进行全面挖掘与融合，并灵活适配各种任务与场景下的稀疏或不完整的轨迹输入。

本文旨在构建时空轨迹的自监督学习方法，解决自监督学习应用于时空轨迹数据时所面临的挑战，实现能够应用于多种下游任务并提升任务准确率的时空轨迹挖掘模型。具体而言，文本围绕时空轨迹中的访问时间信息、上下文信息和行程信息，构建有效挖掘并融合多方面信息的时空轨迹自监督学习方法；同时，针对稀疏或不完整的轨迹输入，构建面向通用性的轨迹自监督学习方法。所构建的方法的有效性在多种轨迹数据集和下游任务上得到了验证。本文提出的方法旨在解决时空轨迹数据中的挑战，以实现更准确的任务处理。

本文的主要研究内容和技术贡献如下：

1）围绕时空轨迹中的访问时间信息，研究了**访问时间感知的轨迹自监督学习方法**，提出了一种时间感知的地点嵌入模型 TALE。该模型提出了一种新颖的时间

哈夫曼树结构，将轨迹点按照访问时间分配到时间节点中，从而提取地点的功能特征。同时，提出了一种时间软划分机制，以减少信息丢失，并提高访问时间信息建模的准确性。在四个轨迹数据集和三种下游任务上的实验验证了 TALE 模型自监督学习的有效性。

2）围绕时空轨迹中的上下文信息，研究了**上下文感知的轨迹自监督学习方法**，提出了一种上下文与时间感知的地点嵌入模型 CTLE。该模型提出了一种考虑轨迹中动态上下文信息的自监督学习方法，能够建模多功能地点的准确功能特征。同时，提出了一个时间编码模块和小时掩码自监督目标，同时考虑了轨迹的两方面访问时间信息。在两个轨迹数据集和轨迹预测下游任务上的实验验证了 CTLE 模型自监督学习的有效性。

3）围绕时空轨迹中的行程信息，研究了**行程信息建模的轨迹自监督学习方法**，提出了一种起终点先验的轨迹行程生成模型 IGOP。该模型首先设计了一种对轨迹噪音鲁棒、有利于模型区分轨迹异常值的轨迹行程表示形式。随后，提出了一种基于扩散生成的自监督学习方法，能够建模轨迹行程与起终点的关联性，并能生成起终点对应的行程。在两个轨迹数据集和两种下游任务上的实验验证了 IGOP 模型自监督学习的有效性和可解释性。

4）针对多方面时空信息的全面挖掘与融合，研究了**多视图融合的轨迹自监督学习方法**，提出了一种轨迹多视图最大熵嵌入模型 MMTEC。该模型通过一个离散轨迹编码器和一个连续轨迹编码器，分别建模能概括轨迹中多方面信息的离散、连续时空视图。同时，提出了一种轨迹的多视图最大熵编码自监督框架，能够有效地融合轨迹的两个视图。在两个轨迹数据集和三种下游任务上的实验验证了 MMTEC 模型自监督学习的有效性和全面性。

5）针对稀疏或不完整轨迹输入的灵活适配，研究了**面向通用性的轨迹自监督学习方法**，提出了一种通用轨迹模型 GTM。该模型将轨迹特征划分为能够独立掩盖与生成的三个特征域，使得模型能够灵活适配各种任务中的不完整轨迹输入。同时，设计了恢复稀疏轨迹对应稠密轨迹的自监督学习目标，能够维持模型在面对稀疏轨迹时的性能。在两个轨迹数据集和三种下游任务上的实验验证了 GTM 模型应对各任务和场景的有效性。

关键词：时空数据挖掘；时空轨迹数据；自监督学习；预训练

ABSTRACT

Spatial-temporal trajectories, which are typically represented as sequences of timestamped locations, capture the travel and movement behaviors of individuals and vehicles within a geographic area. These trajectories play a crucial role in a variety of applications in Intelligent Transportation Systems (ITS), enabling tasks such as location classification, movement prediction, traffic flow prediction, transportation demand estimation, arrival time estimation, traffic anomaly detection, and trajectory-user link analysis. The increasing prevalence of devices capable of recording spatial-temporal trajectories, such as smartphones and vehicle positioning systems, has contributed to the availability of large-scale trajectory data. On the other hand, the growing demand for ITS and smart city applications necessitates the development of comprehensive and effective models for analyzing trajectory data.

Traditional machine learning algorithms rely on manual feature engineering to implement data pattern mining. However, this kind of approaches is costly and struggles to effectively extract complex features from spatial-temporal trajectories. In recent years, numerous studies have demonstrated that neural networks and deep learning methods can automatically and effectively extract features from data. Most existing deep learning methods for spatial-temporal trajectory mining adopt an end-to-end learning approach. However, this solution depends on large-scale labeled datasets, which are often difficult to obtain, limiting its feasibility in practical applications. In contrast, self-supervised learning aims to extract supervisory information from unlabeled datasets, which are generally more readily available. The models trained through self-supervised learning exhibit better generalizability and can adapt to a variety of downstream tasks. Applying self-supervised learning techniques to spatial-temporal trajectory data mining eliminates the reliance on manual feature engineering and large-scale labeled data, and enhances the computational and storage efficiency when applying spatial-temporal trajectory data to various downstream tasks. Therefore, self-supervised learning methods for spatial-temporal trajectories are gradually becoming a key research area that promotes the development of trajectory data mining technology and applications. However, the complexity and specificity of spatial-temporal trajectories pose multiple challenges to the construction of self-supervised learning models. Specifically, self-supervised learning methods need to effectively mine and integrate diverse information from spatial-temporal trajecto-

ries and flexibly adapt to the varied trajectory inputs of different scenes and tasks.

This dissertation aims to develop a self-supervised learning method for spatial-temporal trajectories, addressing the challenges faced when applying self-supervised learning to spatial-temporal trajectory data, and realizing a spatial-temporal trajectory mining model that can be applied to a variety of downstream tasks and improve task accuracy. Specifically, the dissertation revolves around the visit time information, contextual information, and itinerary information in spatial-temporal trajectories, constructing a self-supervised learning method for spatial-temporal trajectories that effectively mines and integrates diverse information. At the same time, for the varied trajectory inputs of different scenes and tasks, a trajectory self-supervised learning method aimed at a general model is constructed. The effectiveness of the constructed method has been validated on a variety of trajectory datasets and downstream tasks.

The main contributions of this dissertation is summarized as follows.

1) Regarding the visited time information within spatial-temporal trajectories, the research on **visited time-aware trajectory self-supervised learning** proposes a time-aware location embedding model named TALE. TALE introduces a novel time Huffman tree structure that allocates trajectory points into time nodes based on their access time, thereby extracting functional features of locations from the visited time information. Additionally, TALE suggests a soft temporal partition mechanism to minimize information loss during visited time partitioning, thereby enhancing the accuracy of modeling information. Experimental validations across four trajectory datasets and three downstream tasks demonstrate the effectiveness of TALE's self-supervised learning.

2) Regarding the contextual information within spatial-temporal trajectories, the research on **contextual-aware trajectory self-supervised learning** introduces a context-aware location embedding model named CTLE. CTLE proposes a self-supervised learning approach considering the dynamic contextual environment of target locations within trajectories, capable of accurately modeling multifunctional location features. Furthermore, CTLE introduces a time encoding module and Masked Hour self-supervised target, considering both absolute visited time information and relative visited time difference information within trajectories. Experimental validations on two trajectory datasets and the trajectory prediction downstream task confirm the effectiveness of CTLE's self-supervised learning.

3) Regarding the itinerary information within spatial-temporal trajectories, the re-

search on **trajectory self-supervised learning for itinerary information modeling** introduces a trajectory itinerary correlation-based origin-destination generation model named DOT. DOT initially introduces a trajectory itinerary representation robust to trajectory noise, aiding the model in distinguishing trajectory outliers. Moreover, it proposes a self-supervised learning method based on diffusion generation for modeling trajectory itinerary correlation, capable of capturing the correlation between trajectory itineraries and origin-destination pairs. Experimental validations across two trajectory datasets and two downstream tasks verify the effectiveness and interpretability of DOT's self-supervised learning.

4) To effectively integrate multiple the diverse information in spatial-temporal trajectories, the research on **trajectory self-supervised learning for multi-view fusion** introduces a multi-view maximum entropy trajectory embedding model named MMTEC. MMTEC models two views of trajectories — travel semantics, and continuous spatial-temporal information — through discrete and continuous trajectory encoders, respectively. It further proposes a multi-view maximum entropy coding self-supervised framework for trajectories, comprehensively integrating the two views of trajectories. Experimental validations across two trajectory datasets and three downstream tasks confirm the effectiveness and comprehensiveness of MMTEC's self-supervised learning.

5) For the flexibly adaptation of sparse or incomplete trajectory input, the research on **general-model-oriented trajectory self-supervised learning** introduces a general trajectory model named GTM. GTM divides trajectory features into three domains that can be masked and generated independently, allowing the model to flexibly adapt to incomplete trajectory input in diverse tasks. At the same time, a self-supervised learning objective for recovering dense trajectories from sparse ones was designed, maintaining the model's performance when dealing with sparse trajectories. Experimental validations across two trajectory datasets and three downstream tasks validated the model's effectiveness in addressing various tasks and scenarios.

KEYWORDS: spatial-temporal data mining, spatial-temporal trajectory data, self-supervised learning, pre-training

目录

1 绪论

时空轨迹是由（地点，时间）对组成的序列，记录了车辆的移动或个人的出行行为。随着地点记录设备的广泛普及，以及基于地点的服务的日益流行，由车载导航、智能手机等设备记录的轨迹数据的可用性和可获取性日益增强。丰富的时空轨迹数据支撑了广泛的时空数据挖掘任务，包括未来轨迹预测[1–3]，到达时间估计[4–7]，异常检测[8–10]，以及轨迹聚类[8,11–13] 等。

为了在上述任务中高效地利用时空轨迹数据，需要开发相应的模型对轨迹数据进行有效的分析。传统的机器学习算法，如线性回归和随机森林等，依赖于高成本的人工特征工程，并且难以有效地挖掘时空轨迹中的复杂信息。基于神经网络的深度学习算法被证明能够自动从时空轨迹数据中捕获特征与信息，摆脱了对人工特征工程的依赖且能提升下游任务的性能。然而，现有面向时空轨迹的深度学习算法大多基于端到端学习方案，依赖大规模的标签数据来取得最佳性能。此类数据的获取通常成本较高甚至不可行，阻碍了端到端深度学习算法在实际应用场景中的部署。同时，端到端学习算法的泛化性能通常较差，难以在不同下游任务间迁移，降低了其计算与存储的效率。自监督学习方案，旨在从无标签数据中提取监督信息，无需高成本的标签数据即可在多种下游任务上取得较好的性能，进而提升深度学习算法在实际应用场景中的可用性与效率。因此，为了将轨迹数据高效地应用于数据挖掘任务，在不依赖大规模标签数据的前提下提高下游任务的泛化性能和准确率，时空轨迹的自监督学习具有较高的研究与应用潜力。然而，时空轨迹数据的特征复杂且多样，这些特征中蕴含多方面的信息，包括访问时间信息、上下文信息、行程信息等。有效挖掘并融合这些信息对于构建适用于多种下游任务的时空轨迹自监督学习方法至关重要。同时，自监督学习模型需要灵活适配不同下游任务和场景涉及到的多样轨迹输入，特别是稀疏或不完整的轨迹输入，来强化其在实际应用场景中的可迁移性和通用性。因此，如何面向时空轨迹的复杂特征、多方面信息和多样输入，设计有效的自监督学习方法，是一个尚存众多挑战的问题，具备较高的研究价值。

1.1 研究背景与意义

时空轨迹是地理科学、环境科学和智能交通等领域常用的概念，是对实体在地理空间中移动过程的采样记录结果，反映了移动实体在时间和空间中的运动路径。图 1-1 展示了一条包含 6 个采样点的时空轨迹，其中每个采样点均为一个（地

点，时间）对，表示实体在移动过程中特定时刻所处的具体地点。随着现代智能交通系统和智慧城市的发展，时空轨迹数据的重要性日益凸显。主要原因在于时空轨迹的研究和应用有助于人们理解和预测在给定时空环境下的复杂现象。以交通场景为例，交通系统涉及大量的移动实体，如车辆、人群等。智能交通系统通过多种采样方式记录实体的运动过程，积累了丰富的轨迹数据。对这些数据的深度挖掘和分析，可以帮助人们从中洞察出各种深层次的信息和规律，如城市活动模式、交通状况、人群行为等，从而优化城市规划、提高交通效率、促进经济社会的持续发展。同时，随着数据收集技术的进步和智能设备的广泛应用，人们可以从多种来源获取到大量的轨迹数据，如卫星定位系统、手机信号、基于地点的服务等。这些数据包含多样的附加业务信息，具有极高的研究价值。

图 1-1　一条时空轨迹的示例

Figure 1-1 Example of a spatial-temporal trajectory.

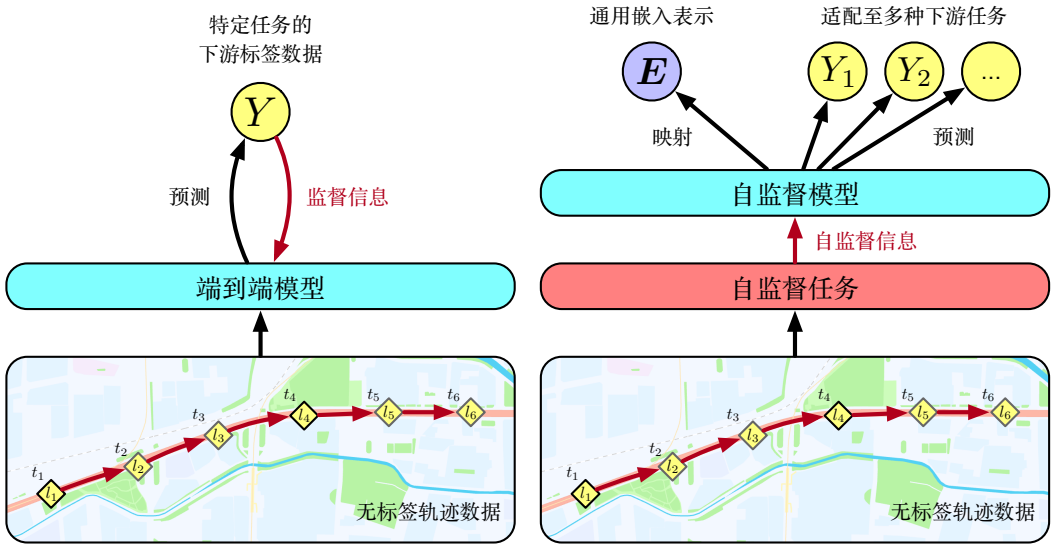
时空轨迹数据能够用于众多富有实用价值的任务，如地点分类、未来轨迹预测、异常行为检测、流量预测、到达时间估计等。这些轨迹数据挖掘任务的开展离不开机器学习与深度学习技术。轨迹数据挖掘的关键在于从轨迹中提取多方面的信息，以图 1-1 中的轨迹为例，能够挖掘的信息包括地点的功能特征、实体的移动模式和路径偏好、路段的交通状况等。

传统的机器学习技术已经被广泛应用于轨迹数据挖掘相关的多种任务，例如决策树算法[14–17] 用于地点分类和流量预测，k 近邻算法[18–20] 用于异常行为检测，最短路径算法[21–23] 用于到达时间估计等。然而，传统机器学习算法通常依赖人工特征工程来达到最佳性能，且受限于模型容量，这类算法对轨迹数据中的复杂信息的建模能力较弱。

近年来，深度学习技术迅猛发展，在自然语言处理、计算机视觉和生物医学等领域取得了突破性的进展。相比传统机器学习模型，深度学习模型无需繁琐的特征工程，能够从数据集中自适应学习特征与任务目标的关联性。受此启发，研

究人员开始将深度学习技术推广到轨迹数据挖掘领域，面向各种下游任务设计了一系列深度学习模型，并研究如何进一步提升深度学习模型在轨迹数据挖掘上的准确率、泛化性能和计算效率等指标。

深度学习方法通常由包含一组可学习参数的神经网络构成，这组可学习参数在随机初始化后，在训练阶段使用反向传播、随机梯度下降等技术进行更新。训练完成后的神经网络可被应用于下游任务。形式化上，神经网络模型可被抽象为函数 f_θ，其中 θ 为可学习参数。在下游任务应用中，f_θ 接收特定对象作为输入，作为映射函数将输入映射为嵌入向量[24–26]，或是直接输出下游任务期望的预测值。为了从数据中提取监督信息，对可学习参数 θ 进行训练，主流的方案可分为**端到端学习**和**自监督学习**。



(a) 端到端学习　　(b) 自监督学习

图 1-2　端到端学习与自监督学习的对比

Figure 1-2 Comparison between end-to-end learning and self-supervised learning.

在轨迹数据挖掘领域，端到端学习[27]方案实现方面的简单性使其成为最常用的模型训练方案。这种方案将模型的各个组件，包括地点嵌入层、轨迹编码器、预测模块等一同通过下游任务的标签数据训练，如图 1-2(a) 所示。然而，在实际应用场景中，端到端方案高度依赖大规模、高质量的标签数据集来获得良好的性能，但此类数据的获取成本高昂，导致端到端方案的可用性欠佳。同时，端到端方案训练的模型缺乏泛化性和可迁移性，带来了额外的存储和运算负担，降低了轨迹数据挖掘任务的效率。为了解决端到端学习的上述问题，研究人员转向基于自监督学习[28]方案的模型。自监督方案通过从可用性较高的无标签轨迹数据集中提取下游任务无关的自监督信息来训练模型参数，如图 1-2(b) 所示。这种做法缓解了对

大规模标签数据的依赖，并且能够学习到具备高泛化性、高可迁移性的模型，进而提升运算效率。同时，设计合理的自监督学习方法能够有针对性地挖掘轨迹数据中的关键信息，进而提升下游任务的准确率。

考虑到时空轨迹数据需要同时支持多种下游任务，且时空轨迹的标签数据通常难以获得，自监督学习模型的下游任务泛化性能和不依赖于标签数据的特性使得其在时空轨迹数据挖掘中具有较高的应用潜力。同时，自监督学习技术在自然语言处理[29,30]和计算机视觉[31,32]领域已经取得了巨大成功。尽管如此，自监督学习在时空轨迹挖掘领域的应用依然面临许多挑战。相较于数据形式相对单一、规整的自然语言和计算机图像，时空轨迹数据通常由多种特征构成，每种特征包含独特的特性。例如，在图 1-1 所示的轨迹中，轨迹的时间特征 t_i 记录了轨迹点发生的时间，拥有周期性、间距不等的特性；轨迹的地点 l_i 由经纬度、路段下标在内的多种特征描述，具有空间异质性、近邻相关性等特性。同时，时空轨迹数据中又蕴含了丰富的信息，包括地点和路网的功能信息、车辆的行驶速度与方向、路段的平均通行时间等。另一方面，多种任务与场景中可能会涉及多样的轨迹输入，特别是稀疏或不完整的轨迹输入。因此，如何合理地构建时空轨迹自监督学习模型，全面挖掘多方面时空信息，灵活地适配稀疏或不完整的轨迹输入，是一个亟待解决的问题。

本文面向时空轨迹数据的自监督学习，针对轨迹数据中的多种特性和多方面信息，提出了一系列自监督学习方法，并将这些方法适配至多种轨迹数据挖掘下游任务，验证了它们的实用性与有效性。具体而言，本文围绕访问时间感知的轨迹自监督学习、上下文感知的轨迹自监督学习、行程信息建模的轨迹自监督学习、多视图融合的轨迹自监督学习和面向通用性的轨迹自监督学习五大内容开展研究，并提出了相应的自监督学习方法，得到了一系列面向时空轨迹数据的自监督模型，弥补了现有方法对信息建模不全面、在下游任务上效果不佳，以及难以迁移至多种下游任务的问题。

1.2 时空轨迹自监督学习的关键挑战

如上文所述，本研究旨在实现时空轨迹的自监督学习模型，以构建适用于多项下游任务的时空轨迹挖掘方法。为实现这一目标，必须克服几个关键挑战，其中包括如何从时空轨迹的复杂特征中提取潜在的多方面信息，如何有效地整合这些多方面信息；以及如何设计一个自监督学习模型，对稀疏或不完整的轨迹输入进行灵活适配。本节将对这些主要挑战进行详细讨论。

1.2.1 多方面时空信息的全面挖掘与融合

多项时空轨迹相关任务需要全面挖掘轨迹中的多方面信息。例如，准确的访问流量预测任务需要对轨迹中的地点功能特征进行建模。然而，时空轨迹数据的特征通常具有复杂性，包括多样性、非线性以及时空依赖性等特点。这些复杂性给时空轨迹自监督学习模型的构建带来挑战，要求模型能够有效地从多样复杂的特征中提取多方面信息。以下详细列出了时空轨迹所涵盖的多方面信息。

(1) 访问时间信息

在日常生活中，人们通常在适当的时间访问具备特定功能特征的地点。例如，于工作日上班时间访问办公楼，于饭点访问餐馆，于傍晚前往公园慢跑或散步，于深夜返回家中休息。相对应的，地点在轨迹中被访问的时间则可以体现出地点的功能特征。图 1-3 展示了纽约市民对具有三种不同功能特征的地点的访问时间分布。可以看到，地点的功能性与其在轨迹中被访问时间的分布具有较强的关联性。因此，轨迹的绝对访问时间信息能够体现地点的功能特征。

轨迹的绝对访问时间信息也能体现地点的多功能性和地点之间的关联性。图 1-4 给出了多位用户的时空轨迹示例。在图中，三位用户分别在 t_1、t_2 和 t_4 时访问地点 l_1，这意味着 l_1 可能是一个多功能地点。同时，地点 l_2 只在 t_3 时有用户造访，这意味着 l_2 可能是一个单功能地点。另外，地点 l_2 和 l_3 在同一时间 t_3 被用户访问，这意味着它们可能具有相似的功能。

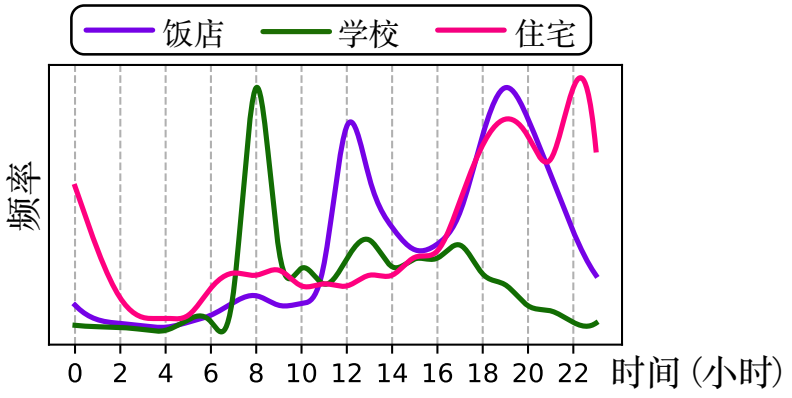


图 1-3　市民对三类地点在不同时间上的访问频率分布

Figure 1-3 The visit frequency distributions of three types of locations by citizens vary over time.

另一方面，轨迹的相对访问时间差信息能体现地点的访问频率和停留时间特征，如图 1-5 所示，在一条时空轨迹中，同一个地点的两次访问之间的时间差体现

了地点的访问频率，而相邻地点之间的访问时间差体现了地点的停留时间。这些信息也能进一步反映地点的功能特征，以及用户对地点的访问偏好。

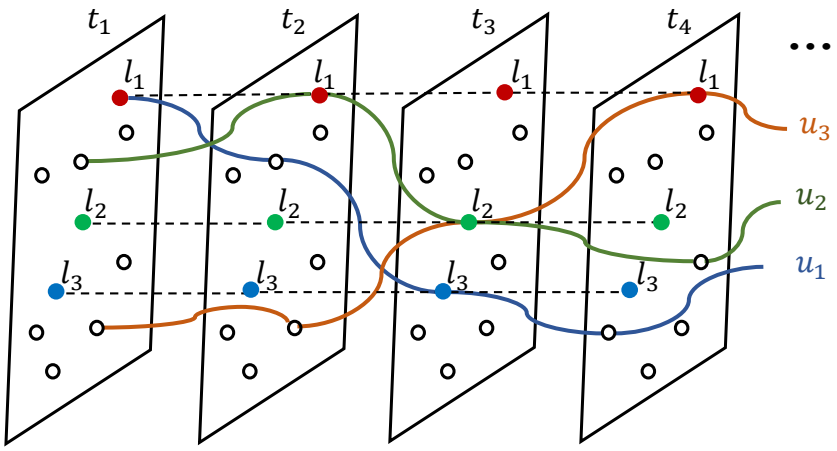


图 1-4　多个用户的时空轨迹示例

Figure 1-4 Example of multiple users' spatial-temporal trajectories.

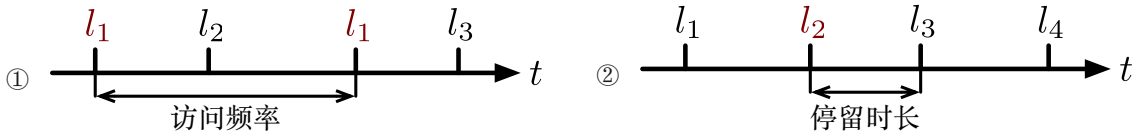


图 1-5　相对访问时间差所揭示的信息

Figure 1-5 Information revealed by relative visited time differences.

(2) 上下文信息

在时空轨迹中，某个轨迹点代表的访问记录同时会伴随着上下文信息，即该轨迹点附近的其他访问记录。现实世界中用户对地点的访问具有顺序性，而这种顺序性对应的上下文信息能够从时空轨迹中挖掘出来。以图 1-6 中的两条轨迹为例，人们通常进行以学生宿舍-知行大厦-学生食堂以及学生宿舍-科技大厦-学生食堂两种顺序访问各个地点。这种顺序性反映出知行大厦、科技大厦两个地点在轨迹中拥有相近的上下文，也就是说人们通常在访问这两个地点的前后访问相似的地点。因此，可以推断这两个地点拥有相近的功能特征。

另一方面，多功能地点在现实世界中也很常见。例如，一座购物中心可能包含餐厅或电影院，一栋办公楼可能包含娱乐场所或健身房。人们会在不同的上下文环境中出于不同的目的访问同一地点，换言之，轨迹中动态的上下文信息能够更准确地体现多功能地点具体的功能特征，如图 1-7 所示。

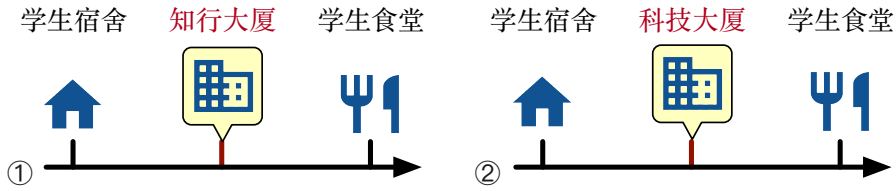


图 1-6　轨迹上下文信息所体现的地点功能特征

Figure 1-6 Location's functionality derived from contextual information in trajectories.

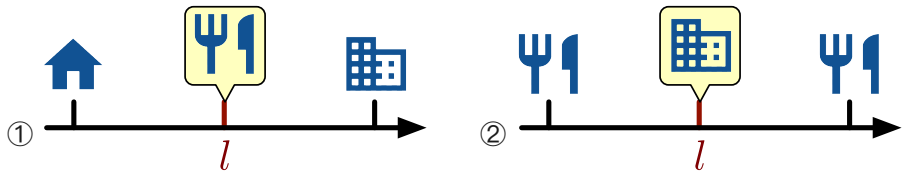
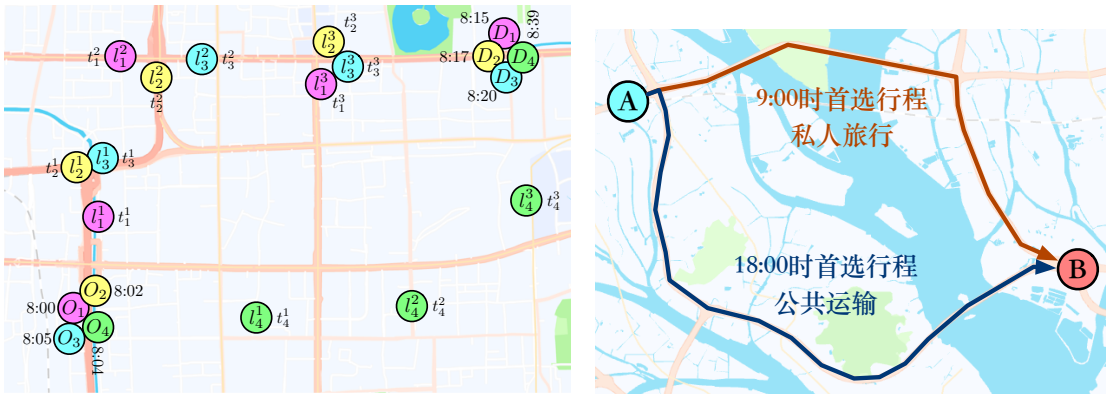


图 1-7　同一地点可能在不同的上下文中承担不同的功能

Figure 1-7 One location may undertake different functionalities in dissimilar contextual neighbors.

(3) 行程信息

时空轨迹是记录一次移动或旅行行为的数据，它代表了从一个地点到另一个地点的整个过程。这些数据由一系列通过定位系统采集的经纬度点组成，这些点往往包含一定的噪声。而这背后所代表的行程，是一个更为抽象的概念，比如“一辆车从出发点经过特定路段，穿越城市某些区域，最终到达目的地”。轨迹的众多特征，如路线选择、行驶距离、旅行时间等，都与其背后的行程密切相关。以旅行时间为例，图 1-8(a) 中有四对拥有相近起点和终点的轨迹，其中下标表示轨迹编号，上标表示轨迹中的地点编号。图中 (O_1,D_1)、(O_2,D_2) 和 (O_3,D_3) 对应的轨迹



(a) 行程与旅行时间的关联性　　(b) 行程与路线偏好的关联性

图 1-8　轨迹中的行程信息所体现的多方面关联性

Figure 1-8 The multiple aspects of correlations reflected by the itinerary information in trajectories.

实际上拥有相似的行程，因此旅行时间非常接近；而 (O_4, D_4) 对应的行程与其他行程不同，因此其轨迹的旅行时间也差异较大。此外，对于相同的起点和终点，由于受到时间段、交通状况等因素的影响，不同的行程可能会选择不同的路线。这种差异反映了在特定条件下的路线偏好，如图 1-8(b) 所示。

在挖掘出时空轨迹的多方面信息后，如何有效融合这些信息，构建出能够满足多种下游任务信息需求的自监督学习模型，是另一项关键挑战。这要求自监督学习模型不仅要能够处理多种类型的信息，还要能够理解不同信息之间的关联性。

1.2.2 稀疏或不完整轨迹输入的灵活适配

时空轨迹数据在真实世界的应用中呈现多样化，不同的应用场景和下游任务可能会有不同形式和质量的轨迹输入，特别是稀疏或不完整的轨迹输入。为了保证时空轨迹自监督学习模型在各种场景和任务中的通用性，如何设计出一个能够灵活适配多样化轨迹输入的自监督学习模型，是实现时空轨迹自监督学习的又一挑战。下面详细阐述了两种特殊的时空轨迹输入，以及它们给时空轨迹自监督学习模型带来的挑战。

(1) 稀疏轨迹输入

时空轨迹通常由离散采样的、带时间戳的地点组成，轨迹的采样间隔则定义为相邻采样点的时间戳之差。在现实世界情景中，稀疏轨迹（即采样间隔较大的轨迹）较为常见[33,34]，因为采样设备的限制或是为了节省存储空间。例如，假设某大城市所有出租车一天的轨迹原始记录的采样间隔为 2 秒，大约需要 200GB 的存储空间。而如果以 4 分钟的采样间隔存储稀疏轨迹，只需要约 1.5GB 的存储空间。然而，从这样的稀疏轨迹中提取轨迹信息是具有挑战性的。

为了说明这一点，考虑图 1-9，其中稠密轨迹 $\mathcal{T} = \langle (l_1,t_1),(l_2,t_2),\ldots,(l_7,t_7)\rangle$ 的采样间隔为 30 秒，如使用 90 秒的采样间隔对其进行重采样，则得到其对应的稀疏轨迹 $\mathcal{T}' = \{(l_1,t_1),(l_4,t_4),(l_7,t_7)\}$。稀疏轨迹输入存在的挑战有如下三个方面：

1）**粗糙和不完整的时空信息**。在稀疏轨迹中，连续点之间的间隔增大，使得提取时空信息变得困难，无法以稠密轨迹相同的细节和准确性进行提取。例如，在图 1-9 中，车辆在点 (l_2,t_2) 和 (l_3,t_3) 之间减速，但这一信息无法直接从稀疏轨迹中提取出来。

2）**错误或不确定的路径选择**。在稀疏轨迹中，采样点在路网上不连续，导致移动对象在两点间的路径选择难以确定。在图 1-9 中，由于稀疏轨迹的稀疏性引起

的不确定性，确定移动对象在点 (l_4,t_4) 和 (l_7,t_7) 之间通过了路段 e_1 还是 e_2 是有挑战性的。

3）**轨迹恢复的效果和效率低下**。可以使用轨迹恢复方法[35–37] 将稀疏轨迹恢复为稠密轨迹。然而，这种做法需要引入一种二阶段方案，即使用额外的轨迹挖掘方法从恢复的轨迹中获取信息。这种方案会导致误差累积，降低下游任务的准确性，同时对计算效率产生负面影响。

图 1-9　一条稠密轨迹和其对应的稀疏轨迹

Figure 1-9 A dense and its corresponding sparse trajectory.

(2) 不完整的轨迹输入

时空轨迹支撑的下游任务多种多样，其中一些下游任务在输入阶段无法提供完整的轨迹特征。图 1-10 中给出了两种不完整输入轨迹的下游任务示例：

1）**轨迹的旅行时间估计**：对于旅行时间估计任务，存在一种情形是需要估计一段尚未发生的行程的旅行时间，而此行程仅有出发地点、目的地点和出发时间在预测阶段是已知的。换言之，可以认为该任务的输入仅包含轨迹的起点、终点和起点时间戳。

2）**轨迹预测**：对于轨迹预测任务，通常已知的信息仅是一条完整行程轨迹的已发生部分，需要对未发生的轨迹序列以及目的地进行预测。

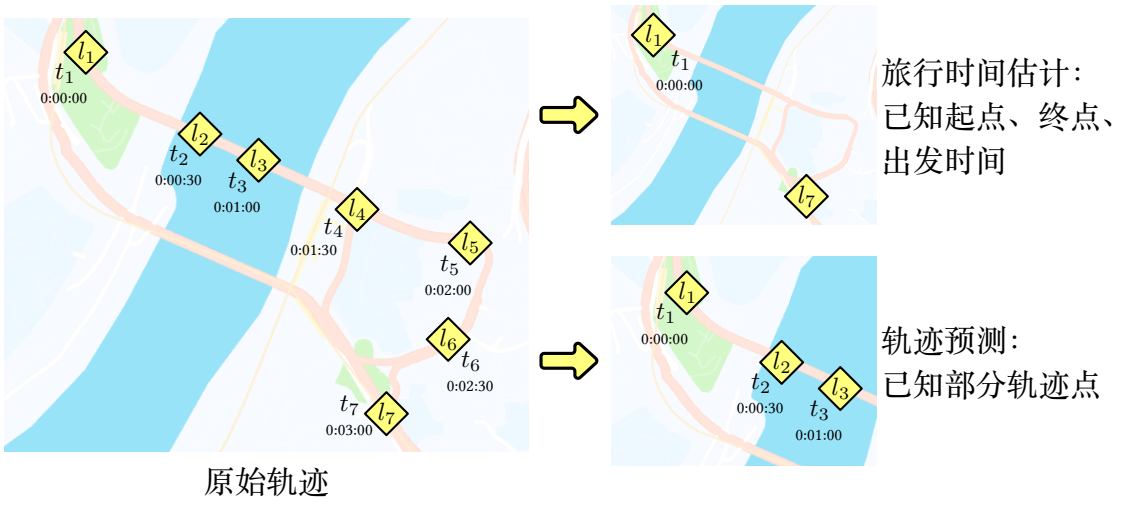


图 1-10　两种下游任务的不完整轨迹输入

Figure 1-10 The incomplete trajectory input of two downstream tasks.

常见的时空轨迹数据挖掘模型通常对轨迹序列的完整性和序列中每个轨迹点的特征的完整性有较高的要求，然而这将限制它们能应用的下游任务的范畴。

为了让时空轨迹自监督学习模型能够有效适应稀疏或不完整的轨迹输入，需要模型具备良好的泛化能力和可调整性，能够通过预处理或模型内部的自适应机制，处理不同特性的轨迹数据，以适应不同的下游任务需求。

1.3　国内外研究现状

目前国内外面向时空轨迹数据的自监督学习，针对多方面时空信息全面挖掘与融合的挑战，展开了初步探索。本节将对国内外相关研究现状进行分析和总结。

1.3.1　基于词嵌入模型的时空轨迹自监督学习

在自然语言处理[29,38]领域中，通过自监督学习得到词的嵌入向量是一种常见做法。这种做法将自然语言语料库中的每个词映射为一定长度的嵌入向量，随后使用自监督学习的辅助任务对嵌入向量进行训练。自监督词嵌入方法的核心思想是，具有相似上下文环境的词的含义通常相似，因此应当拥有相近的嵌入向量。word2vec[38]是自监督词嵌入模型的代表之一，它通过最大化目标词和上下文共同出现的概率来实现自监督学习辅助任务。

实际上，时空轨迹与自然语言中的句子有一定的共性。如第 1.2.1 节所述，在时空轨迹中拥有相似上下文环境的地点具有相似的功能特征，因此时空轨迹包含

和自然语言类似的上下文关联性信息，如图 1-11 所示。因此，许多时空轨迹的自监督学习模型将自监督词嵌入模型的做法迁移到轨迹数据上。例如，DeepMove[39] 应用 Skip-gram[38] 来建模轨迹中的上下文信息，而 Yao 等[40] 则基于 N-gram 模型[41] 来建模轨迹中的访问序列性。

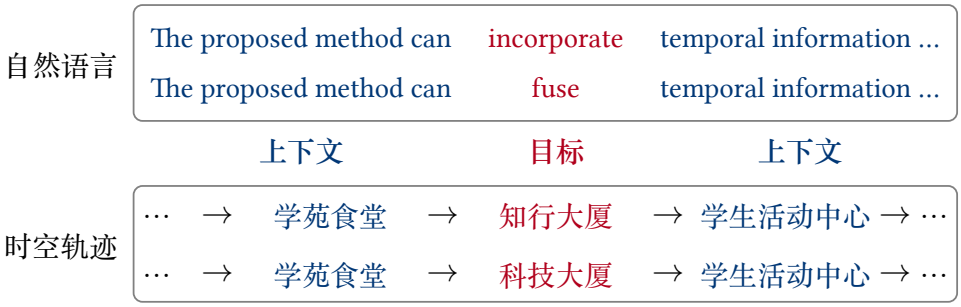


图 1-11　时空轨迹与自然语言的上下文关联性的相似之处

Figure 1-11 The similarity of contextual correlations between spatial-temporal trajectories and natural language.

然而，时空轨迹数据相较于自然语言中的句子，还包含一些独特的信息。如第 1.2.1 节所述，从轨迹访问时间信息中能够提取出更全面的地点功能特征。因此，自监督学习模型应该考虑访问时间信息。Geo-teaser[42] 通过扩展 Skip-gram 中的输出嵌入向量来区分在工作日和周末被访问的地点。POI2Vec[43] 在 word2vec 框架的基础上考虑了地点间的空间相关性。HIER[44] 通过向 N-gram 模型提供额外的时间嵌入向量来建模访问时间信息。现有工作与实验表明，时间信息确实有助于构建更有效的自监督学习方法。然而，现有工作对时空轨迹中的访问时间信息挖掘并不全面。首先，现有工作并未考虑细粒度级的访问时间分布所体现的地点功能特征。其次，现有工作忽略了相对访问时间差所体现的地点访问频率和停留时间信息。

另一方面，如第 1.2.1 节所述，对于具有多种功能的地点，需要模型能从轨迹的上下文信息中准确地建模地点的特定功能。自然语言处理领域也面临类似的挑战，即需要更准确地表示有多种含义的词在特定上下文环境中的含义。近年来，以 ELMo[45] 和 BERT[29] 为首的自监督上下文词嵌入模型有效地解决了多义词嵌入的问题。这些方法的核心思路在于根据上下文环境，动态地计算单词的嵌入向量。然而，现有基于词嵌入模型的时空轨迹自监督学习模型仅为每个地点分配一条嵌入向量，这意味着它们无法区分多功能地点的特定功能。

1.3.2 基于生成模型的时空轨迹自监督学习

生成模型，包括序列到序列（seq2seq）模型、对抗生成网络[46] 和扩散模型[47] 等，通常以特定特征为先验条件，输出给定先验条件下符合特定分布的数据。自监督生成模型的核心思想在于，通过自监督辅助任务训练生成模型，使得模型能够从生成的数据的分布中提取关键性的特征。

自编码器（Auto-encoder）[48] 是一种典型的自监督生成模型，其由两个主要部分组成：编码器（Encoder）和解码器（Decoder），其中编码器负责将输入数据映射到一个隐藏的特征空间（也称为编码空间），这个过程通常涉及到数据维度的降低；解码器负责将这个隐藏空间中的表示重新映射回原始数据空间，尽可能恢复输入数据的原貌。自编码器的自监督辅助任务通过最小化解码器还原的原始数据的差异构建。解码器能够以较高的准确率从隐藏空间中的表示还原原始数据，意味着编码器映射的隐藏空间表示中包含关于原始数据的关键信息。自编码器在计算机视觉（Computer Vision, CV）领域被广泛用于构建图像自监督学习模型[49]。现有时空轨迹自监督学习方法中有许多基于自编码器模型，学习时空轨迹的低维嵌入向量。Trembr[50] 在编码和恢复中考虑轨迹中的路网信息和访问时间信息，有效地提取轨迹的路网相关特征。t2vec[51] 和 trajectory2vec[11] 将地理空间划分为网格，并使用自编码建模网格化的轨迹来挖掘轨迹的空间信息。TrajectorySim[52] 通过特征工程从轨迹原始特征中计算速度、加速度、方向等特征，并作为自编码器的输入和还原对象，有效地建模轨迹中的高阶特征。然而，自编码器强调对于输入数据中蕴含的信息的建模，而现有方法均将轨迹中的单方面信息以特征提取的方式抽象为自编码器的输入特征，未全面建模轨迹中多方面的信息。

自监督生成模型也可以仅由单个生成网络构成，不涉及原始输入数据的压缩与还原。此种生成模型旨在根据输入的条件信息，直接生成下游任务所需要的生成目标，如根据图像的部分特征输出完整的图像[53]。自监督学习的辅助任务基于最大化生成的准确率构建，并使生成模型的训练不依赖于大规模的标签数据。自监督生成模型在时空轨迹上的典型应用之一是起终点行程生成方法。这些方法根据输入的起点、终点特征，生成行程对应的轨迹序列，并在时空轨迹数据集上以自监督的方式训练，以此建模轨迹行程与起终点间的关联性。基于传统算法的行程生成方法，如 Dijkstra 算法[21] 和各种多样性路由方法[54–59] 侧重于输出起终点对应的最小旅行成本行程。然而，这些方法生成的行程可能与实际的轨迹行程不符，也无法直接从轨迹数据中建模轨迹行程关联性。DeepST[60] 通过建模历史轨迹数据，从而提高了生成的准确性，但它无法很好地处理历史轨迹中的噪音和异常值。

1.3.3 基于对比模型的时空轨迹自监督学习

对比学习模型[32,61]在近年来成为机器学习领域的一个研究热点，特别是在表示学习中显示出了巨大的潜能。这类方法通过构造一个对比损失（Contrastive Loss）来引导模型学习，使得相似的样本在表示空间中更加接近，而不相似的样本则相距较远。具体来说，模型被训练以将正样本对（通常是同一实体的两个不同视图或变体）拉近，同时将负样本对（通常是来自两个不同实体的样本）推远。该类方法的核心优势在于能够有效地利用未标注数据，从而在大规模数据集上学习到具有鲁棒性的数据表示。

近年来，许多工作基于自监督对比学习模型构建时空轨迹的表示学习模型。PreCLN[62]通过结合对比学习和轨迹的空间、路网特征，为轨迹预测任务提供更全面的信息。SML[63]引入了对比预测编码框架[61]，以自监督的方式构建了一种移动行为学习方法。START[64]提出了一种轨迹自监督学习方法，结合掩码语言模型[29]和基于 SimCLR 损失函数[32]的对比学习训练任务来增强其学习能力。然而，现有工作大多基于对比学习对时空轨迹中的某方面信息进行建模，未对自监督对比学习模型进行扩展来全面地建模轨迹中的多方面信息。同时，现有工作也并未讨论多方面信息的有效融合问题，无法保证学习到的轨迹表示在多种下游任务上能获得一致的性能。

1.4 研究内容与主要贡献

本文旨在解决自监督学习应用于时空轨迹数据时所面临的多种挑战，并构建有效建模并融合时空轨迹中多方面信息、能够灵活适配多样轨迹输入的自监督学习方法。所构建的方法能够将时空轨迹数据同时应用于多种下游任务，并提升下游任务的泛化性能和准确性。图 1-12 展示了本文的总体研究框架。本文包含 5 项研究内容，其中前 3 项研究内容围绕构建挖掘多方面信息的时空轨迹自监督学习方法开展，第 4 项研究内容关注有效融合挖掘到的多方面信息，第 5 项研究内容关注灵活适配稀疏或不完整的轨迹输入，强化时空轨迹自监督学习模型的适配能力。

本文具体开展的研究内容和贡献总结如下：

1）围绕时空轨迹中的访问时间信息，开展**访问时间感知的轨迹自监督学习**相关研究，提出了一种时间感知的地点嵌入模型 TALE。TALE 提出了一种新颖的时间哈夫曼树结构，将轨迹点按照访问时间分配到时间节点中，进而从访问时间信息中提取地点的功能特征。同时，TALE 提出了一种时间软划分机制，减少访问时

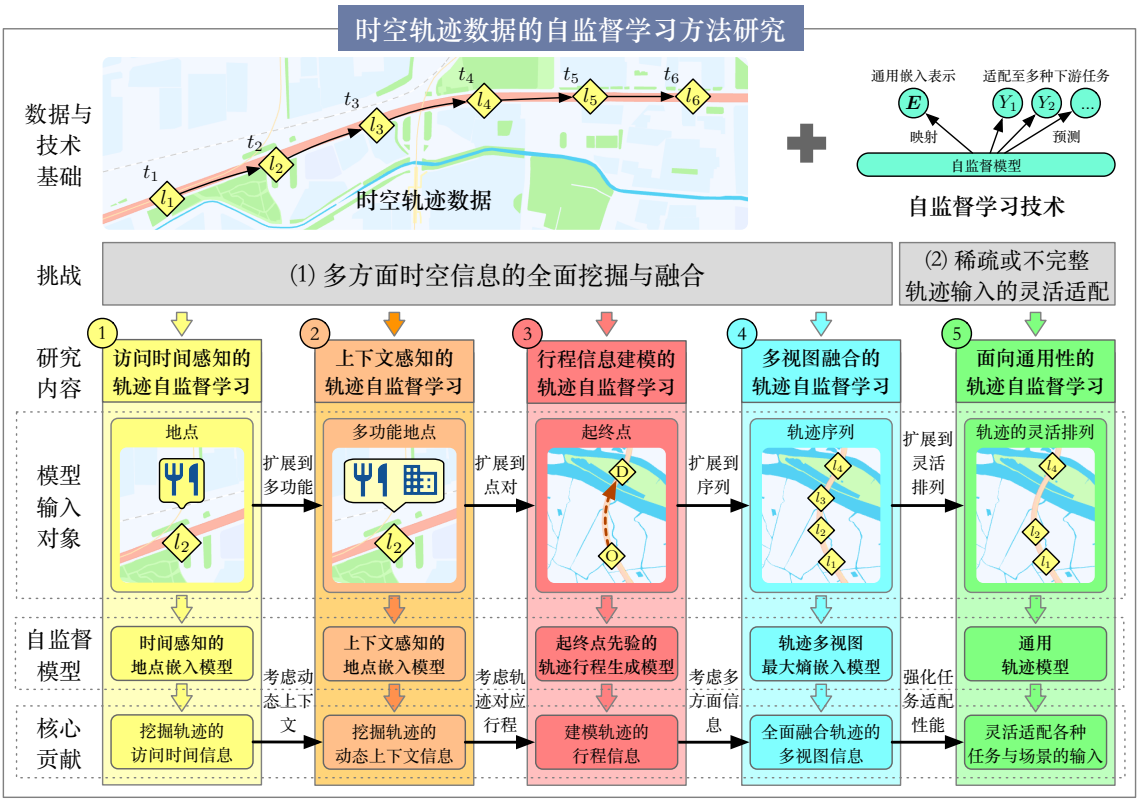


图 1-12　本文研究内容框架

Figure 1-12 The framework of the research contents.

间划分过程中的信息丢失，提升访问时间信息建模的准确性。在三个签到记录数据集和一个手机信令数据集，以及地点分类、访问流量预测、轨迹预测三种下游任务上的实验，证明了 TALE 模型的有效性。

2）围绕时空轨迹中的动态上下文信息，开展**上下文感知的轨迹自监督学习**相关研究，提出了一种上下文与时间感知的地点嵌入模型 CTLE。CTLE 提出了一种考虑轨迹中目标地点动态上下文环境的自监督学习方法，能够建模多功能地点的准确功能特征。同时，CTLE 提出了一个时间编码模块和小时掩码自监督目标，同时考虑了轨迹的绝对访问时间信息和相对访问时间差信息。在两个手机信令数据集和轨迹预测下游任务上的实验，验证了 CTLE 模型的有效性。

3）围绕时空轨迹中的行程信息，开展**行程信息建模的轨迹自监督学习**相关研究，提出了一种起终点先验的轨迹行程生成模型 IGOP。IGOP 首先设计了一种对轨迹噪音鲁棒、有利于模型区分轨迹异常值的轨迹行程表示形式，并提出了一种基于扩散生成的行程信息建模的轨迹自监督学习方法，能生成起终点对应的轨迹行程。在两个出租车定位数据集和起终点行程生成、起终点旅行时间估计两个下游任务上的实验，验证了 IGOP 模型的有效性和可解释性。

4）针对多方面时空信息的全面挖掘与融合，开展**多视图融合的轨迹自监督学习**相关研究，并提出了一种轨迹多视图最大熵嵌入模型 MMTEC。MMTEC 将轨迹中的多方面信息归总为离散、连续时空两个视图，并通过一个离散轨迹编码器和一个连续轨迹编码器分别将两个视图建模为嵌入向量。随后，提出了一种轨迹的多视图最大熵编码自监督框架，能够有效地融合两个视图。在两个出租车定位数据集和轨迹预测、相似轨迹搜索、轨迹旅行时间估计三个下游任务上的实验，证明了 MMTEC 模型的有效性和全面性。

5）针对稀疏或不完整轨迹输入的灵活适配，开展**面向通用性的轨迹自监督学习**相关研究，并提出了一种通用轨迹模型 GTM。GTM 将时空轨迹的特征划分为时间、空间、路段三个可独立掩盖和自回归生成的特征域，使得模型能够灵活地适配各种任务的不完整轨迹输入。同时，GTM 构建了基于稀疏轨迹还原稠密轨迹的自监督学习目标，能够在应对稀疏轨迹输入时保持较好的性能。在两个出租车定位数据集和轨迹预测、轨迹恢复、起终点旅行时间估计三个下游任务上的实验，证明了 GTM 对各任务和场景的适配能力。

1.5 论文组织框架

本文以时空轨迹数据的自监督学习研究为主题，面向时空轨迹中的多方面信息，设计自监督表示模型，解决轨迹自监督学习中的关键挑战。论文包含 8 章内容，具体组织如下：

第 1 章为本文的绪论。首先，介绍了本文的研究背景和意义，阐述了时空轨迹数据的特性，以及自监督表示学习问题的重要性。随后，列举了时空轨迹数据中的关键性信息。接下来，介绍了时空轨迹自监督学习的国内外研究现状。最后，概述了本文的研究内容与主要贡献，并展示了本文的组织框架。

第 2 章介绍本文的相关定义与理论基础。首先，对文中设计的多种数据进行形式化定义，包括路网结构、空间地点、起终点和时空轨迹。随后，介绍了实验中使用的三种轨迹数据集，包括签到记录数据集、手机信令数据集和出租车定位数据集。最后，系统性地介绍了文本的基础性研究框架——自监督学习，并对时空轨迹的自监督学习相关挑战、步骤和下游任务进行了分析。

第 3 章介绍访问时间感知的轨迹自监督学习研究内容及其对应的时间感知的地点嵌入模型 TALE。提出新颖时间哈夫曼树结构和时间软划分机制，将轨迹的访问时间信息融入自监督学习中。在四个轨迹数据集和三种下游任务上进行实验验证了模型的有效性。

第 4 章介绍上下文感知的轨迹自监督学习研究内容及其对应的上下文感知的

地点嵌入模型 CTLE。提出考虑轨迹动态上下文环境的自监督学习方案，以准确建模多功能地点的特定功能特征。在两个轨迹数据集和轨迹预测任务上进行实验验证了模型的有效性。

第 5 章介绍行程信息建模的轨迹自监督学习研究内容及其对应的起终点先验的轨迹行程生成模型 IGOP。提出一种自监督训练的起终点先验的轨迹行程生成模型，建模轨迹行程与起终点的关联性。在两个轨迹数据集和两种下游任务上进行实验验证了模型的有效性。

第 6 章介绍多视图信息融合的自监督学习研究内容及其对应的轨迹多视图最大熵嵌入模型 MMTEC。该模型将轨迹中的多方面信息归总为两个视图，结合所提出的轨迹最大熵编码自监督学习框架，能有效建模并融合轨迹的两个视图，实现对轨迹多方面信息的有效融合。在两个轨迹数据集和三种下游任务上进行实验验证了模型的有效性。

第 7 章介绍面向通用性的轨迹自监督学习研究内容及其对应的通用轨迹模型 GTM。该模型将时空轨迹划分为三个可独立生成的域，并通过自监督学习强化了对稀疏轨迹输入的鲁棒性，增强了模型适配稀疏或不完整轨迹输入的能力。在两个轨迹数据集和三种下游任务上进行实验验证了模型的有效性。

第 8 章对本文的研究工作和贡献进行总结，并展望未来的研究工作。

2 相关定义与理论基础

本文就时空轨迹数据的自监督学习开展研究，旨在将自监督学习技术应用于时空轨迹数据挖掘，有效地建模并融合时空轨迹中的多方面信息，发挥自监督学习的优势以构建可迁移至多种下游任务的时空轨迹自监督学习模型。本章将对时空轨迹数据相关的概念进行形式化定义，介绍实验中所使用的时空轨迹数据集，阐述自监督学习的基本概念、主要方法，并分析时空轨迹自监督学习的主要挑战、一般步骤和下游任务。

2.1 时空轨迹相关定义

时空轨迹数据通常由对象在地理空间中的移动行为产生，数据以带时间戳的地点序列为形式表示与存储。本节将对时空轨迹数据的相关概念进行形式化定义，包括数据的发生场景路网、数据的组成单元地点、轨迹序列和轨迹的起终点。

2.1.1 路网

道路网络[65,66]是支撑交通系统和现代化社会运转的核心要素，也是地理信息系统中的基础对象。在地理信息系统中，一般用道路网络描述和模拟城市、地区或全球范围内的道路系统。它包括站点、小区、路口、卡口等空间地点要素，也包括道路、街道、高速公路等线性地理要素，以及这些要素之间的连接关系。本研究中的路网结构是对现实世界中道路网络的抽象表示。

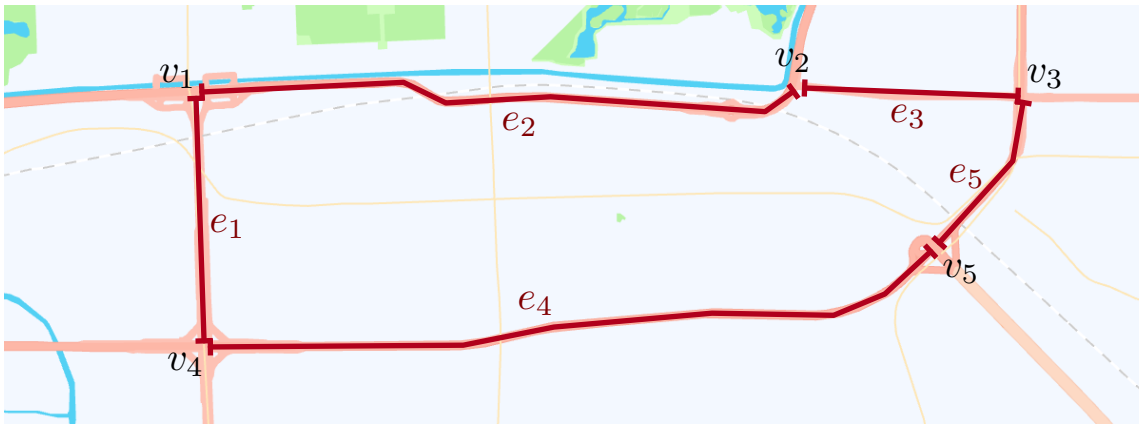


图 2-1 包含 5 个节点和 5 条路段的路网

Figure 2-1 A road network containing five nodes and five road segments.

定义 2.1 (路网，Road Network)**.** 一个路网被建模为一个有向图 $\mathcal{G}=(\mathcal{V},\mathcal{E})$，其中 $\mathcal{V}$ 是一组节点，每个节点 $v_i \in \mathcal{V}$ 表示道路段之间的交叉口或路段的末端，$\mathcal{E}$ 是一组边，每条边 $e_i \in \mathcal{E}$ 表示连接两个节点的路段。一条边由起始节点和结束节点定义：$e_i=(v_j,v_k)$。图 2-1 展示了一个路网的示例。

2.1.2 空间地点

图 2-2 三个路网上的空间地点

Figure 2-2 Three spatial locations on the road network.

空间地点[67,68]是地理信息系统中的一种静态组成部分，它们不仅仅是地球表面上的特定区域，更是一种具有特定属性和含义的地点。这些地点可能是由经纬度定义的地球表面上的地点，可能是建筑物、城市、自然景观等拥有明确功能性的兴趣点，也可能是由路段和距离比例定义的路网上的地点，如图 2-2 所示。

定义 2.2 (GPS 地点，GPS Location)**.** 一个被 *GPS* 坐标定义的地点 l 是一对经纬度定义的地球表面上的特定地点：$l=(l^{\text{lng}},l^{\text{lat}})$，$l^{\text{lng}}$ 和 l^{lat} 分别表示经度和纬度。

定义 2.3 (POI 地点，POI Location)**.** 兴趣点（*Point of Interest, POI*）通常指地理空间中具备明确功能特征的区域或建筑，例如加油站、百货大楼、住宅楼、医院、公园、饭店等。在一个地理信息系统中，通常将所有的 *POI* 构成集合 $\mathbb{P}$，其中每个 *POI* 地点 l 由其在集合中的下标表示。

定义 2.4 (路网地点，Road Network Location)**.** 一个在路网上的地点 l 由路段和路段上前进的比例共同定义：$l=(e,r)$，其中 $e \in \mathcal{E}$ 是该地点所在的路段，r 是该地点在路段上前进距离相对全长的比例。

2.1.3 时空轨迹

时空轨迹数据由一系列带时间戳的地点访问记录构成。这些数据通常用于记录移动对象（如人、车辆、动物等）在空间中的运动轨迹。时空轨迹数据中的轨迹

点反映了对象在不同采样时刻所处的地点。通过时空轨迹数据，可以了解对象的移动路径、速度、停留时间等信息。

定义 2.5 (时空轨迹，Spatial-temporal Trajectory)**.** 一条时空轨迹 $\mathcal{T}$ 是一系列带有时间戳的地点：$\mathcal{T} = \langle (l_1,t_1),(l_2,t_2),\ldots,(l_{|\mathcal{T}|},t_{|\mathcal{T}|})\rangle$，其中 l_i 可以是符合定义 *2.2* 的第 i 个地点的空间坐标，也可以是符合定义 *2.3* 的第 i 个地点的 *POI* 下标。时间戳 t_i 表示访问 l_i 的时间。

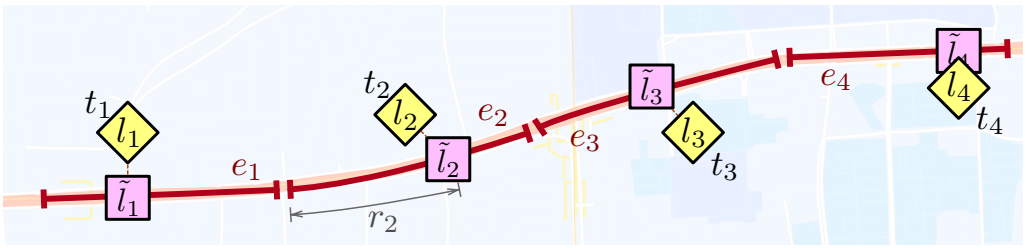


图 2-3　一条轨迹以及其对应的路网匹配轨迹

Figure 2-3 A trajectory and its map-matched counterpart.

由符合定义 2.2 的地点构成的时空轨迹一般发生在路网上。应用地图匹配算法[69]，可以将轨迹 $\mathcal{T}$ 投射到路网 $\mathcal{G}$ 上。得到的结果即所谓的地图匹配轨迹（Map-matched Trajectory），表示为 $\widetilde{\mathcal{T}}$。

定义 2.6 (地图匹配轨迹，Map-matched Trajectory)**.** 地图匹配的轨迹可以表示为 $\widetilde{\mathcal{T}} = \langle (\widetilde{l}_1,t_1),(\widetilde{l}_2,t_2),\ldots,(\widetilde{l}_{|\mathcal{T}|},t_{|\mathcal{T}|})\rangle$，其中 $\widetilde{l}_i = (e_i,r_i)$ 是符合定义 *2.4* 的路网地点。

示例 2.1. 图 *2-3* 给出了一条示例轨迹 $\mathcal{T} = \langle (l_1,t_1),(l_2,t_2),(l_3,t_3)(l_4,t_4)\rangle$，其中地点用菱形表示。$\mathcal{T}$ 对应地图匹配轨迹 $\widetilde{\mathcal{T}} = \langle (\tilde{l}_1,t_1),(\tilde{l}_2,t_2),(\tilde{l}_3,t_3),(\tilde{l}_4,t_4)\rangle$，其中地点用方形表示。例如，$l_2$ 被匹配到 $\tilde{l}_2$，它位于道路段 e_2 的中间。因此，使用 r_2 来表示车辆已行驶过的 e_2 长度的比例，从而得到 $\widetilde{l}_2 = (e_2,r_2)$。

由于轨迹均为对移动对象地点的离散性采样，在轨迹的采样和存储时涉及到轨迹的采样间隔。

定义 2.7 (轨迹采样间隔，Sampling Intervals of Trajectories)**.** 轨迹的采样间隔 η 是指相邻点之间的时间间隔，即 $\eta = t_i - t_{i-1}$，其中 $i \in \{2,3,\ldots,|\mathcal{T}|\}$。对于一条原始采样间隔较短的稠密轨迹，可以重采样得到其稀疏版本，用 μ 来表示重新采样后的稀疏轨迹的采样间隔。

2.1.4 起终点

在交通系统中，市民和车辆一般会频繁地进行从某个起始地点出发，到达某个终止地点的行为。这种行为通常伴随着较强的旅行语义，如为了上班从住宅区域前往办公楼，为了就餐从办公楼前往餐馆。这里将这样的起点和终点构成的点对作为一种特殊的概念进行研究，称之为起终点。在交通工程、城市规划、流行病学、旅游学研究等现实世界场景中，都需要开展起终点分析，为各种实际问题提供解决方案和策略建议。例如，精确地理解与预测个体或群体从起点到终点的移动模式，能够揭示人们的日常习惯和偏好，并能为交通流量预测、城市资源分配和传染病传播模型提供宝贵的信息。

定义 2.8 (起终点，Origin-Destination (OD) Pair)**.** *起终点定义为某次旅行行为中的出发地点 l_o 和目标地点 l_d 构成的有序元组 (l_o, l_d)，如图 2-4 所示。其中两个地点 l_o 和 l_d 可以是符合定义 2.2、2.3 或 2.4 的地点。*

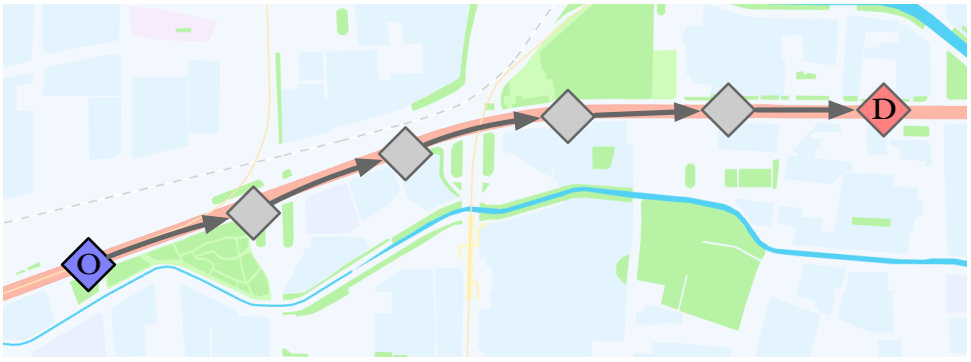


图 2-4 一段旅程的起终点

Figure 2-4 Origin-destination pair of a trip.

2.2 时空轨迹数据集

一定空间和时间范围内采集的时空轨迹可整合为一个时空轨迹数据集，记录了对应时空范围内多个对象的移动行为。本节对论文中所使用的轨迹数据集的来源和内容进行介绍，并介绍各数据集的统计指标。本文中所使用的时空轨迹数据按照来源可分为三大类：签到记录数据集、手机信令数据集和出租车定位数据集。

2.2.1 签到记录数据集

签到（check-in）指的是基于地点的服务平台（如国内的大众点评和国外的Foursquare）中的用户在到达某个兴趣点后，在平台上对自己的本次访问进行标记的行为。每次签到都将产生一个包含符合定义 2.3 的 POI 地点和签到时间戳的记录。某位用户一定时间内的签到记录按照时间排列，即构成了一条时空轨迹。

本文的相关实验使用了三个签到记录数据集，分别是美国-纽约、日本-东京和印度尼西亚-雅加达的 Foursquare 签到数据 [1]，记为 Foursquare-NYC、Foursquare-TKY 和 Foursquare-JKT。在数据预处理中，总访问记录数量少于 10 的用户和少于 5 的地点被舍弃。经过预处理后的数据集统计信息如表 2-1 所示。

表 2-1 签到记录数据集统计
Table 2-1 Statistics of check-in record datasets.

数据集	用户量	地点数量	轨迹点数量
Foursquare-NYC	1,077	3,908	82,091
Foursquare-TKY	2,290	7,057	389,063
Foursquare-JKT	9,193	13,105	536,792

2.2.2 手机信令数据集

在现代手机的蜂窝网络架构中，运营商部署的每个信号基站负责为附近一定范围内的手机用户提供信号，而手机用户在信号基站之间的切换将被运营商记录为信令数据。每个基站可被看作是一个 POI 地点，基于信令数据可以还原用户对各个基站的访问行为。因此，手机信令数据对应业务背景中的时空轨迹的形式化定义和签到记录数据类型业务背景中的基本一致。两者主要的区别在于手机信令数据集中的轨迹点采样密度比签到记录数据集更高，能够更完整地表示用户对各地点的访问行为。

本文的相关实验使用了两个手机信令数据集，分别为北京和沈阳信令数据，记为 Mobile-PEK 和 Mobile-SHE。在数据预处理中，首先移除平均停留时间低于 30 分钟的基站和用户，因为其相关的轨迹点绝大多数均为路过的地点。随后，保留包含超过 5 个停留点的轨迹，以及拥有超过 4 条轨迹的用户。经过如此预处理后的数据集的统计信息如表 2-2 所示。

[1] https://www.kaggle.com/datasets/rishabhchandra/foursquare-complete-dataset

表 2-2　手机信令数据集统计
Table 2-2 Statistics of mobile signaling datasets.

数据集	用户数量	地点数量	轨迹点数量	时间跨度
Mobile-PEK	12,691	7,279	1,383,422	5 天
Mobile-SHE	10,564	7,201	607,581	11 天

2.2.3　出租车定位数据集

一般情况下，现代出租车业务信息系统从每一个订单的开始到结束，会利用车辆上的定位系统或是司机的手机定位功能，以一定的时间间隔对车辆所处地点进行记录。每次记录会产生一个包含时间戳和符合定义 2.2 的地点的轨迹点，每个订单的所有轨迹点按时间排列即构成一条时空轨迹。

本文的相关实验使用了成都、西安的网约车定位数据[2]，以及哈尔滨[4]、葡萄牙波尔图的出租车定位数据。同时，为了能应用地图匹配算法以得到如定义 2.6 所描述的地图匹配轨迹，本文实验中从 OpenStreetMap（OSM）[70] 上下载了各城市的路网。完成地图匹配后，路网中至少被一条轨迹所覆盖的路段会被保留。表 2-3 给出了预处理后的数据集的统计信息。

表 2-3　出租车定位数据集统计
Table 2-3 Statistics of taxi location datasets.

数据集	轨迹数量	路段数量	轨迹点数量
成都	298,995	3,791	7,025,468
西安	376,407	3,558	10,198,837
哈尔滨	614,830	3,704	12,692,468
波尔图	55,120	2,225	1,482,751

上面介绍的三种数据集均是对地理空间中对象移动行为的记录，同时由于产生方式与场景的不同，它们也拥有不同的特性。签到记录数据集中的时空轨迹每个轨迹点均为用户对地点明确的访问行为，因此轨迹的采样间隔相对较长，且不一定是对用户完整移动行为的记录。手机信令数据集由手机用户在基站服务区之间的转移记录构成，轨迹的采样间隔依然较长，但对用户的移动行为记录相对完整。出租车定位数据集由定位系统在车辆行驶过程中对位置的高频率采样形成，因此

[2]https://gaia.didichuxing.com/

轨迹的采样间隔较短，是对车辆移动行为的高精度记录。不同类型数据集的不同特性使得它们所适用的场景、任务，以及适合的建模方案均有所差异。

2.3 自监督学习基础

本研究旨在基于自监督学习技术，从时空轨迹数据中挖掘多方面信息，并构建能够迁移至多种下游任务、具备任务通用性的时空轨迹自监督学习模型。本节将围绕自监督学习框架，介绍自监督学习的基本概念、主要方法，以及时空轨迹的自监督学习。

2.3.1 自监督学习的基本概念

自监督学习（self-supervised learning）是一种机器学习范式，它利用数据本身的结构来学习数据的表示，而不是依赖于外部的标签或者人工的输入。这种方法试图通过自生成的监督信号来训练模型，以此达到从大量未标记数据中自动提取有用信息的目的。自监督学习在深度学习、特征学习、模式识别等领域有着广泛的应用。

自监督学习的核心思想是利用数据本身的结构性（如时序性、空间性等）来生成监督信号，以此引导学习过程。换句话说，它通过构建一个辅助任务（pretext task）来预测数据中的某些部分或属性，这一过程不需要人工标注的数据。通过这种方式，模型能够学习到数据的内在特征和结构，从而获得更好的数据表示。

自监督学习的实施过程可以划分为几个关键步骤，这些步骤共同构成了自监督学习框架的核心。以下是自监督学习的一般步骤：

1）**定义辅助任务（Pretext Task）**：辅助任务是自监督学习中的关键概念，它是指在没有外部标签指导下，通过数据本身生成的任务。这个任务的设计应当能够促使模型学习到数据的有用特征。辅助任务的选择依赖于数据的类型和特点，以及期望模型学习到的特征。常见的辅助任务包括图像的部分重建、未来帧预测、文本中的词语填空等。

2）**数据预处理与增强**：在定义了辅助任务后，需要对数据进行适当的预处理和增强，以生成用于自监督学习的数据集。这可能包括裁剪、旋转图像或对文本数据进行遮蔽处理等操作。数据增强不仅可以提高模型对数据变化的鲁棒性，还可以扩大训练数据集，有助于模型学习到更丰富的特征表示。

3）**构建自监督模型**：根据辅助任务的性质选择合适的模型架构。这一步骤通

常涉及到选择一个合适的神经网络架构，如卷积神经网络（Convolutional Neural Network, CNN）[71] 用于处理图像数据，循环神经网络（Recurrent Neural Network, RNN）[72] 或 Transformer[73] 用于处理序列数据。模型的设计应当能够有效地从辅助任务中学习到数据的内在特征。

4）**设计损失函数**：损失函数用于计算模型输出与自监督信号之间的差异，是模型训练过程的关键。在自监督学习中，损失函数需要能够准确反映辅助任务的目标，促使模型学习到有用的数据表示。例如，在一个图像重建任务中，损失函数可能是重建图像与原图之间的像素级差异。

5）**模型训练与优化**：利用构建的自监督模型和设计的损失函数，通过反向传播算法对模型进行训练。在这个过程中，模型参数会根据损失函数的梯度进行更新，以最小化损失。模型训练通常需要大量的计算资源，并且可能需要调整超参数，如学习率、批大小等，以获得最佳的训练效果。

6）**特征提取与下游任务**：一旦自监督模型训练完成，可以利用其学到的特征表示来执行下游任务，如分类、检测或其他任何需要利用这些特征的任务。通常，模型的一部分（例如，CNN 的卷积层）会被用作特征提取器，而针对特定下游任务的层（如全连接层）可能会在此基础上进行进一步的训练。

自监督学习在许多数据挖掘领域已经得到了广泛的应用，主要由于其多项优势：

1）**减少标注成本**：自监督学习不依赖于人工标注的数据，可以大幅降低数据准备的成本。

2）**提高模型泛化能力**：通过学习数据的内在结构和特征，自监督学习有助于提高模型的泛化能力。

3）**利用未标记数据**：自监督学习能够利用大量的未标记数据，这对于数据标注成本高昂或难以获取标注的领域尤为重要。

尽管自监督学习具有诸多优势，但在实践中也面临一些挑战，如设计有效的辅助任务、处理大规模未标记数据的计算成本、以及如何评估学习到的特征的质量等。

2.3.2 自监督学习的主要方法

自监督学习的方法多种多样，本节将介绍三种常见的自监督学习方法：自监督词嵌入模型、自监督生成模型和自监督对比学习模型。

(1) 自监督词嵌入模型

词嵌入（word embeddings）[74,75] 是自然语言处理中的重要模型。具体而言，词嵌入模型将单词映射为一组实数值的向量化、固定长度、分布式的稠密表示，用以解释单词的文本含义。

静态词嵌入模型 f_θ 的可学习参数 θ 主要由一个嵌入矩阵 $\boldsymbol{E}^{|\mathbb{W}|\times d}$ 构成，其中 $|\mathbb{W}|$ 是语料库中单词的数量，d 为嵌入向量的维度。给定一个目标词 w，词嵌入模型从嵌入矩阵中取该词对应的行向量并输出为嵌入向量，从而完成从词到嵌入向量的映射。该过程形式化定义为：

$$\boldsymbol{z}_w = f_\theta(w) = \boldsymbol{E}_w, \tag{2-1}$$

其中 $\boldsymbol{z}_w$ 为词 w 的嵌入向量，$\boldsymbol{E}_w$ 表示矩阵 $\boldsymbol{E}$ 的第 w 行。

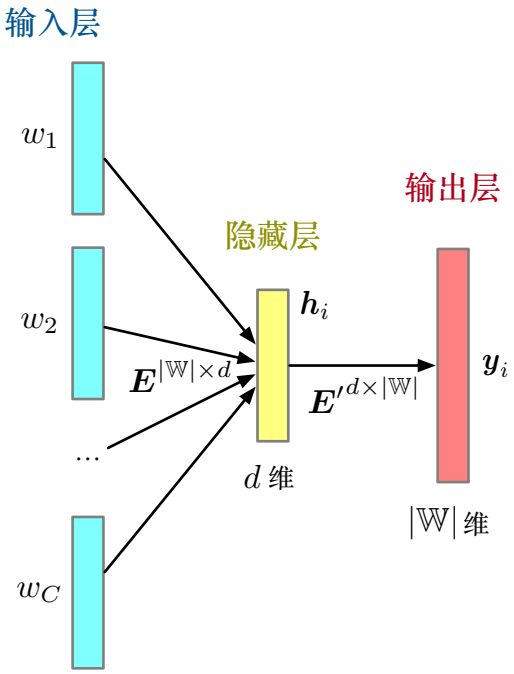


图 2-5　连续词袋模型[76]

Figure 2-5 Continuous bag-of-word model.

为了自监督训练静态词嵌入模型，通常需要设计辅助任务使得具有相似含义的词的在嵌入空间中相近。word2vec[38,76] 是一种经典的自监督静态词嵌入模型，连续词袋（Continuous bag-of-word, CBOW）是它的常见实现方案。图 2-5 展示了 CBOW 在多个上下文词设置下的结构。CBOW 的输入层为公式 (2-1) 中定义的词嵌入模型。计算隐藏向量 $\boldsymbol{h}_i$ 时，CBOW 将输入的上下文单词对应向量取平均，得

到均值向量，并使用输入层将该均值向量映射到隐藏空间中：

$$\begin{aligned} \boldsymbol{h} &= \frac{1}{C}(f_\theta(w_1)+f_\theta(w_2)+\cdots+f_\theta(w_C)) \\ &= \frac{1}{C}(\boldsymbol{z}_{w_1}+\boldsymbol{z}_{w_2}+\cdots+\boldsymbol{z}_{w_C}) \end{aligned} \tag{2-2}$$

随后，CBOW 的辅助任务目标函数定义为：

$$\begin{aligned} \mathcal{L} &= -\log p(w_O|w_1,w_2,\ldots,w_C) \\ &= -\boldsymbol{z}'_{w_O}\boldsymbol{h}^\top + \log \sum_{j'=1}^{|\mathbb{W}|} \exp(\boldsymbol{z}'_{w_j}\boldsymbol{h}^\top), \end{aligned} \tag{2-3}$$

其中 w_O 表示目标词。通过最大化给定上下文情况下目标词出现的概率，CBOW 能够使得训练语料库中频繁出现在类似上下文环境中的词拥有相近的词嵌入向量，从而将词的语义特征体现在它们的嵌入向量中。

上下文词嵌入模型（contextual word embeddings）[45,77] 是静态词嵌入模型的拓展。与仅为每个词分配一条固定嵌入向量的静态词嵌入模型不同，将目标词映射为嵌入向量时，上下文嵌入的映射模型 f_θ 同时接受目标词 w 和其上下文（即以目标词为中心附近的 C 个词组成的集合 $\{w_1,w_2,\ldots,w_C\}$）作为模型的输入，并动态地建模目标词与上下文的关联性。该过程可形式化定义为：

$$\boldsymbol{z}_w = f_\theta(w,\{w_1,w_2,\ldots,w_C\}) \tag{2-4}$$

在语料库中，同一个词通常有多种含义，而词的具体含义可以通过其所处的上下文环境体现。因此，上下文嵌入能够更准确地建模词的含义特征。映射函数 f_θ 需要建模多个词之间的关联性，通常采用循环神经网络[72] 和 Transformer[73] 等序列模型作为基础架构。

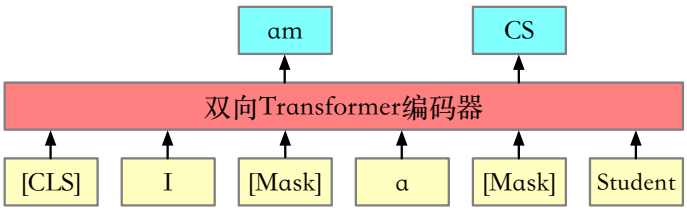


图 2-6　MLM 的构建示例[29]

Figure 2-6 Implementation example of MLM.

为了自监督训练上下文词嵌入模型，通常需要设计辅助任务使得拥有相似上下文环境的词的上下文嵌入向量相近。BERT[29] 模型提出的掩码语言模型（Masked Language Model, MLM）是一种经典的自监督上下文词嵌入模型。在训练过程中，

MLM 会随机掩盖一定比例的输入词，随后通过剩余词提供的上下文来预测被掩盖的单词，辅助任务的训练目标则通过最大化预测的准确率来构建。通过这种训练模式，MLM 能够建立句子中词之间的关联性，并可以针对各种下游自然语言处理任务进行微调。图 2-6 展示了 MLM 的一个构建示例。

(2) 自监督生成模型

深度生成模型[46,78]是一种能够从训练集中学习数据分布，并生成符合该数据分布的样本的模型。扩散模型（Diffusion Models）[47,53]因其训练稳定且生成的样本质量高，成为了近几年来深度生成模型的首选基础架构，特别是在计算机视觉领域。

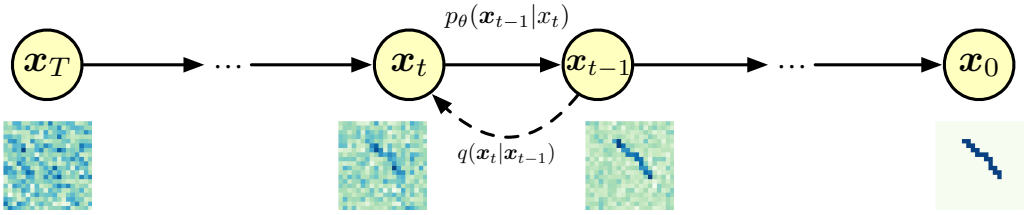


图 2-7　扩散模型的两个马尔可夫过程[47]

Figure 2-7 Two Markov processes of the diffusion model.

扩散模型的核心思想在于将符合标准高斯分布的噪音数据通过逐步去噪过程得到干净的生成数据。如图 2-7 所示，扩散模型包含两个马尔可夫过程：一个是预定义的前向扩散过程 q，过程中逐步向训练样本中添加高斯噪音，直到得到纯噪声为止；一个是可学习的逆向去噪扩散过程，通过训练神经网络 p_θ，从纯噪声中逐步除去噪声，直到得到生成的样本为止。两个过程均由 t 索引，总共进行 T 步。从 $t=0$ 开始，从训练数据集中取一个真实样本 $\boldsymbol{x}_0$，前向扩散过程中的每一步会从高斯分布中采样一定的噪音并加入样本。当 T 足够大且每步添加的噪音大小符合一定的条件时，$\boldsymbol{x}_T$ 将符合标准高斯分布。形式上，前向扩散过程的每一步添加噪音过程可定义为：

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}), \tag{2-5}$$

其中，β_t 由预定义的噪音规划控制，至少满足 $0<\beta_1<\beta_2<\cdots<\beta_T<1$。逆向去噪过程则相反，从标准高斯分布中采样得到噪音样本 $\boldsymbol{x}_T$，每步需预测去除一定噪音后的样本分布，最终得到无噪音的生成样本。形式上，逆向去噪过程的一步可定义为：

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \Sigma_\theta(\boldsymbol{x}_t, t)), \tag{2-6}$$

其中，μ_θ 和 Σ_θ 均为神经网络，给定当前步的噪音样本，预测去噪后样本的均值和误差。

为了自监督训练生成模型，辅助任务通常设计为给定生成模型部分掩盖、替换的数据作为输入，使生成模型还原未被处理的原始数据作为输出，进而强化生成模型对原始数据分布与特征的理解。降噪自编码器（Denoising Autoencoder, DAE）是一种经典的自监督生成模型，旨在通过重建经过人为加噪的输入数据来学习数据的稳健特征表示。通过训练模型以从损坏的输入中恢复原始数据，DAE 学习到的特征表示对于噪声具有较强的鲁棒性。具体而言，降噪自编码器的训练过程包括以下几个关键步骤：

1）**数据加噪**：在训练之前，对原始输入数据添加噪声或是应用其他损坏方案，如添加高斯噪声或是掩盖部分特征等。

2）**还原过程**：损坏的数据输入生成模型，如上面介绍的扩散模型。随后，生成模型输出还原的、无损坏的数据。

3）**计算损失**：通过损失函数计算还原数据和原始未加噪数据之间的差异。常用的损失函数包括均方误差（MSE）和交叉熵等。

4）**反向传播和参数更新**：基于损失函数的结果，通过反向传播算法更新模型的权重和偏置，以最小化重建误差。

降噪自编码器可以学习到数据的稳健特征表示，这些特征可以用于各种下游任务，如分类、聚类等。同时，训练得到的生成模型也直接用于生成数据，或是用于去除数据中的噪声，恢复数据的原始信号。

(3) 自监督对比学习模型

自监督对比学习（Self-supervised Contrastive Learning）是一种在机器学习领域中日益受到重视的学习范式。它通过对比不同数据样本之间的相似性和差异性，来学习数据的有效表示。对比学习的核心思想是，将相似的（正）样本拉近，将不相似的（负）样本推远，从而在特征空间中形成有区分度的数据表示。这种方法在许多领域，如计算机视觉、自然语言处理和语音识别中，已经显示出其强大的性能和潜力。

对比学习的基本原理是通过构建正负样本对，并设计一个目标函数，使得模型能够通过最小化这个目标函数来学习数据的鉴别性特征表示。在这个过程中，模型被训练以区分来自相同分布的正样本对和来自不同分布的负样本对。具体而言，对比学习通常涉及几个关键要素：

1）**正负样本对的构建**：正样本对通常是指同一类别或具有高度相似性的样本对，而负样本对则是指不同类别或具有较低相似性的样本对。在实际应用中，构建正负样本对的策略对学习效果有重要影响。

2）**表示学习**：对比学习的目标是学习一个映射函数，该函数能将原始数据映射到一个特征空间，在这个空间中，正样本对的表示更接近，而负样本对的表示则更远离。

3）**对比损失函数**：对比学习通常采用对比损失（Contrastive Loss）来度量正负样本对之间的相似性和差异性。其中，最著名的对比损失函数包括三重项损失（Triplet Loss）和信息噪声对比估计（InfoNCE）损失等。

在对比学习的要素中，对比损失函数对于对比学习模型的性能至关重要。为了确保对比学习模型在不同的下游任务中具有良好的泛化性，需要尽可能减少对比损失函数引入的偏向性。最大熵的信息论原理可用于衡量对比损失函数的偏向性。具体来说，根据最大熵原理，在明确定义的先验条件下，最能代表系统当前状态的概率分布是具有最大熵的分布[79]。利用这一原理，最大熵编码（Max Entropy Coding, MEC）[80] 旨在在特定先验条件下最大化表示模型的信息熵，以构建偏向性更少的对比损失函数，使得对比学习模型能够适配于多种下游任务。

假设有一组嵌入向量 $\boldsymbol{E} \in \mathbb{R}^{d \times N}$，其中 d 是嵌入向量的维度，N 是向量的数量。为了提供一个可计算的信息熵表达式，MEC 提出了如下的编码长度函数：

$$L = \frac{N+d}{2} \log\det(\boldsymbol{I}_N + \frac{d}{N\varepsilon^2}\boldsymbol{E}^\top \boldsymbol{E}), \tag{2-7}$$

其中，$\boldsymbol{I}_N$ 是维度为 N 的单位矩阵，ε 是解码误差的上界。此函数计算了有损数据编码的编码长度，可以看作是对表示 $\boldsymbol{E}$ 信息熵的估计。在 MEC 框架下，自监督对比学习模型的训练目标为最大化公式 (2-7)。

2.4 时空轨迹的自监督学习

2.4.1 时空轨迹自监督学习的挑战分析

时空轨迹自监督学习是一种结合了时空轨迹数据特性和自监督学习机制的研究领域，旨在从时空轨迹数据中自动提取重要的多方面信息，以驱动众多下游任务（如轨迹分类、轨迹聚类、移动模式识别等）取得良好的性能。考虑到时空轨迹数据通常同时应用于多种下游任务，同时时空轨迹的标签信息难以获取，时空轨迹自监督学习的任务可迁移性和不依赖于标签信息的特性使其具备广泛的应用潜力。尽管如此，时空轨迹自监督学习在实际设计与应用过程中仍面临一系列挑战。

首先，时空轨迹数据通常具有高度的复杂性，包括但不限于特征的多样性、数据的非线性、时空依赖性、多尺度性质以及高维特征空间。这些特性给有效表示学习带来了挑战，要求模型能够从复杂多样的特征中，捕捉到数据在时间和空间上的长期依赖关系及其内在的动态变化模式。

其次，自监督学习的关键之一在于设计有效的辅助任务，以促进有用特征的提取。对于时空轨迹数据而言，如何设计这样的辅助任务尤为关键，需要充分考虑时空数据的特性，设计能够充分挖掘并融合时空轨迹中多方面信息的辅助任务。辅助任务的设计直接影响到学习到的特征的质量和适用性。

最后，在时空轨迹自监督学习中，一个重要的挑战是如何确保构建好的自监督学习模型具有良好的泛化能力，能够适用于不同的下游任务。这不仅要求模型设计和辅助任务设计能够保证自监督学习模型能够全面挖掘多方面信息，还需要模型的设计能够应对时空轨迹不同下游任务与场景的多样化输入形式与输出要求，例如轨迹恢复的不完整的轨迹输入特征，相似性度量的多条轨迹输入，轨迹预测的部分轨迹点输入，以及一些场景下的稀疏轨迹输入。

2.4.2 时空轨迹自监督学习的一般步骤

与第 2.3.1 节中介绍的自监督学习的关键步骤相对应，时空轨迹自监督学习通常也遵循一定的构建步骤。与普适性的自监督学习不同，时空轨迹自监督学习的构建需要考虑到时空轨迹数据的特殊性，并尝试解决上述时空轨迹自监督学习的挑战。下面列举了时空轨迹自监督学习的一般构建步骤。

1）**定义辅助任务（Pretext Task）**：在时空轨迹自监督学习中，首先需要定义一个或多个辅助任务。这些任务应当能够从根本上促进模型捕捉到轨迹数据中的时空特征。例如，预测轨迹中未来某一时刻的位置，或者是给定轨迹片段的排序任务。这些任务都要求模型理解轨迹的时空结构，从而在解决这些任务的过程中学习到有用的时空特征。

2）**数据预处理与增强**：考虑到时空轨迹数据的特殊性，对数据进行适当的预处理和增强是必要的。预处理步骤可能包括轨迹的归一化、去噪和插值等，以确保数据的质量和一致性。数据增强则可能涉及到对轨迹进行随机扰动、缩放或旋转等操作，以增强模型对轨迹变化的鲁棒性，并提高模型学习到的特征的泛化能力。

3）**构建自监督模型**：时空轨迹自监督学习的模型构建需要考虑到轨迹数据的时空特性。通常，可以采用词嵌入模型来对轨迹中地点、路网等元素进行嵌入，或是采用循环神经网络（RNN）、长短期记忆网络（LSTM）或 Transformer 等模型

架构来处理轨迹序列数据。这些模型能够有效地捕捉时序数据中的长期依赖关系，是处理时空轨迹数据的理想选择。模型的设计应当能够从辅助任务中有效地学习到时空轨迹的内在特征。

4）**设计损失函数**：损失函数的设计对于自监督学习至关重要，它直接影响到模型训练的效果。在时空轨迹自监督学习中，损失函数需要能够准确反映辅助任务的目标。例如，如果辅助任务是预测轨迹的未来位置，则损失函数可以是预测位置与实际位置之间的距离。损失函数的选择和设计需要根据具体的辅助任务和数据特性来确定。

5）**模型训练与优化**：随后，通过反向传播算法和选定的优化器对模型进行训练。在训练过程中，模型参数会根据损失函数的梯度进行更新，以最小化损失。此过程可能需要大量的计算资源，同时，可能需要对学习率、批大小等超参数进行调整，以优化训练过程并提高模型性能。

6）**特征提取与下游任务**：一旦自监督模型训练完成，便可以利用其学到的特征表示或是自监督模型本身来执行下游任务。这些任务可能包括轨迹分类、相似轨迹搜索、轨迹预测等。在执行下游任务时，模型的一部分（如神经网络中的某些层）可以被用作特征提取器，而针对特定下游任务的层可能需要在此基础上进行进一步的训练或调整。

2.4.3 时空轨迹自监督学习的下游任务

按照上述时空轨迹自监督学习的一般步骤，最后一步是将自监督学习模型应用于下游任务中。这也是时空轨迹自监督学习的主要应用途径。本节将介绍一系列常见的时空轨迹自监督学习的下游任务。为了描述的便利，本节将时空轨迹自监督学习模型表示为 f_θ，其中 θ 是可学习参数。

地点分类任务： 通常可以根据功能将地点分为多种类型，如餐馆、宾馆、学校、住宅、商城等。该问题可抽象为 $l \to \widehat{Y}$，其中 l 为目标地点，$\widehat{Y}$ 为分类结果。通常将输入对象为地点 l 的自监督模型 f_θ 迁移至此任务，具体的方案有两种：

1）拼接可学习的分类模块：将目标地点 l 作为自监督模型的输入，并拼接一个多分类模块，输出为对该地点的分类概率预测。形式化定义为 $\widehat{Y} = g(f_\theta(l))$。

2）基于最近邻的无监督分类：首先，利用自监督模型 f_θ 将每个地点 l 都映射到嵌入空间中。随后，应用 kNN[81] 等无监督分类算法实现地点分类。目标地点的分类结果由嵌入空间中距离其最近的 k 个训练地点投票决定。

访问流量预测任务： 地点访问流量记录了地点被访问次数随时间的变化情况。地点访问流量的预测除了依赖于历史流量序列外，也依赖于自监督模型所提供的地点特征。该问题可抽象为 $\{l, \langle x_1, x_2, \ldots, x_n \rangle\} \rightarrow \hat{x}_{n+1}$，其中 l 为目标地点，$\langle x_1, x_2, \ldots, x_n \rangle$ 为访问流量序列，$\hat{x}_{n+1}$ 为未来访问流量预测结果。通常将输入对象为地点 l 的自监督模型 f_θ 迁移至此任务。具体而言，对于一个目标地点 l，通过 f_θ 得到其嵌入向量 $\boldsymbol{z}_l$，并将其访问流量序列的每步 x_i 映射到与 $\boldsymbol{z}_l$ 相同的维度，然后将它们拼接起来。连接后得到的特征向量序列作为一个序列预测模块的输入。最后，将预测模块的输出向量送入全连接层来预测未来访问流量值。

起终点行程生成任务： 起终点代表某旅行的出发和目标地点。以导航场景为例，对于一次未来旅行，生成其对应的预测行程能够为出行者提供路线参考。该问题可抽象为 $(l_o, l_d, t_o) \rightarrow \widehat{\mathcal{T}}$，其中 l_o、l_d 和 t_o 分别为出发地点、目标地点和出发时间，$\widehat{\mathcal{T}}$ 为生成的行程对应轨迹。通常将输入对象为起终点 (l_o, l_d) 的自监督模型 f_θ 迁移至此任务。具体而言，通过自监督训练生成模型的方式，使得训练得到的模型 f_θ 能够直接应用于起终点行程生成任务中，即 $\widehat{\mathcal{T}} = f_\theta(l_o, l_d, t_o)$。

起终点旅行时间估计任务： 对于一次未来旅行，估计其旅行时间可服务于外包运输服务中的定价[82,83]、整体出行成本的估算[84]、运输调度[85,86]、送货服务[87,88]和交通流预测[89]等智能交通应用场景。该问题可抽象为 $(l_o, l_d, t_o) \rightarrow \hat{\Delta t}$，其中 $\hat{\Delta t}$ 为估计的旅行时间。通常将输入对象为起终点 (l_o, l_d) 的自监督模型 f_θ 迁移至此任务。具体而言，给定起终点和出发时间 (l_o, l_d, t_o) 作为模型 f_θ 的输入，将其输出作为一个预测模块 g 的输入，并输出估计的旅行时间。形式化定义为 $\hat{\Delta t} = g(f_\theta(l_o, l_d, t_o))$。

相似轨迹搜索任务： 轨迹间存在一定的时空特征相似性，相似轨迹搜索旨在寻找与给定轨迹最相似的候选轨迹，进而应用于轨迹聚类[8,11–13]、异常检测[8–10]、轨迹-用户链接[90,91]等任务中。该问题可抽象为 $\{\mathcal{T}, \mathbb{T}'\} \rightarrow \mathcal{T}'$，其中 $\mathcal{T}$ 为目标轨迹，$\mathcal{T}' \in \mathbb{T}'$ 为最相似的候选轨迹，$\mathbb{T}'$ 为候选轨迹集。通常将输入对象为轨迹 $\mathcal{T}$ 的自监督模型 f_θ 迁移至此任务。具体而言，利用模型 f_θ 将目标轨迹 $\mathcal{T}$ 和每条候选轨迹 $\mathcal{T}'$ 映射为嵌入向量，即 $\boldsymbol{z}_{\mathcal{T}} = f_\theta(\mathcal{T})$，$\boldsymbol{z}_{\mathcal{T}'} = f_\theta(\mathcal{T}')$。随后，基于余弦相似度或欧式距离，将轨迹的所对应嵌入向量的相似度作为轨迹间的相似度，依次得到相似度最高的候选轨迹。

轨迹预测任务： 给定车辆、个人的历史轨迹，可以结合历史轨迹中体现的短期移动模式和轨迹数据集中体现的全局移动模式，预测该车辆、个人的未来轨迹。轨迹预测可用于未来交通状况分析[1]、交通规划[92]、资源调度[62]，或目的地推荐[93]等任务中。该问题可抽象为 $\mathcal{T}_{\text{(his.)}} \rightarrow \hat{\mathcal{T}}_{\text{(fut.)}}$，其中 $\mathcal{T}_{\text{(his.)}}$ 为历史轨迹，$\hat{\mathcal{T}}_{\text{(fut.)}}$ 为预测出的未来轨迹。根据自监督学习的设计以及 f_θ 的输入对象，将 f_θ 适配于轨迹预测任

务的方案可以分为三种：

1）将历史轨迹的地点嵌入序列作为输入特征序列：对于给定的历史轨迹 $\mathcal{T}_{(\text{his.})}$ 和输入对象为地点的模型 f_θ，使用 f_θ 将其中每个地点 l 映射为对应的嵌入向量 $\boldsymbol{z}_l$，并将嵌入向量构成的序列作为序列预测模型的输入，得到预测轨迹。

2）用轨迹映射函数拼接预测模型：对于输入对象为轨迹的嵌入映射模型 f_θ，将历史轨迹 $\mathcal{T}_{(\text{his.})}$ 映射为嵌入向量，并拼接一个预测模型，输出预测轨迹。

3）用生成模型直接轨迹预测结果：对于输入对象为轨迹的生成模型 f_θ，将历史轨迹 $\mathcal{T}_{(\text{his.})}$ 作为 f_θ 的输入，将生成结果作为轨迹预测的结果。

轨迹旅行时间估计任务： 与起终点旅行时间估计不同，该类任务根据给定的不包含时间戳的轨迹 $\mathcal{T}$，对 $\mathcal{T}$ 对应的旅行时间进行估计，可用于基于路径规划的地图应用中。该问题可抽象为 $\mathcal{T} \rightarrow \hat{\Delta t}$，其中 $\mathcal{T}$ 为不包含时间信息的轨迹，$\hat{\Delta t}$ 为估计的旅行时间。通常将输入对象为轨迹 $\mathcal{T}$ 的自监督模型 f_θ 迁移至此任务。具体而言，将 $\mathcal{T}$ 作为模型 f_θ 的输入，并在其输出上拼接预测模块 g，得到旅行时间估计值，即 $\hat{\Delta t} = g(f_\theta(\mathcal{T}))$。

稀疏轨迹恢复任务： 定义 2.7 中介绍的稀疏轨迹在实际数据集中广泛存在，其稀疏性会导致其时空特征精准度下降。稀疏轨迹恢复旨在通过补全稀疏轨迹的中间点，缩短稀疏轨迹的采样间隔，将稀疏轨迹恢复为稠密轨迹。该问题可抽象为 $\mathcal{T}' \rightarrow \widehat{\mathcal{T}}$，其中 $\mathcal{T}'$ 为采样间隔较长的稀疏轨迹，$\widehat{\mathcal{T}}$ 是对应的采样间隔较短的稠密轨迹。通常将输入对象为稀疏轨迹 $\mathcal{T}'$ 的自监督模型 f_θ 迁移至此任务。具体而言，通过自监督训练，使模型 f_θ 在输入稀疏轨迹 $\mathcal{T}'$ 时能生成稠密轨迹 $\widehat{\mathcal{T}}$ 来实现稀疏轨迹的恢复任务，即 $\widehat{\mathcal{T}} = f_\theta(\mathcal{T}')$。

2.5 本章小结

本章首先对时空轨迹的相关概念进行了形式化定义，包括路网结构、空间地点、起终点和时空轨迹。其中，路网结构是时空轨迹产生的主要背景，空间地点是时空轨迹的基本组成部分，起终点描述了时空轨迹的转移语义信息，时空轨迹本身则用于记录地点间转移行为的移动模式。

随后，本章介绍了研究中所使用的时空轨迹数据集，按照数据的来源分为三大类：签到记录数据集、手机信令数据集和出租车定位数据集。其中，签到记录数据集由用户在基于地点的服务平台上签到所产生的记录组成，手机信令数据集记录了移动手机在信号基站之间的转移行为，出租车定位数据集包含出租车在执行订单期间按一定频率采样的地点记录。

最后，本章系统性地介绍了自监督学习的基本概念和主要方法。在普适性自监督学习的基础上，本章还分析了时空轨迹自监督学习的主要挑战和一般步骤，并列举了时空轨迹自监督学习的下游任务。

本文后续的研究内容将基于本节介绍的框架，搭建时空轨迹数据的自监督学习模型，并在各数据集和下游任务上对模型进行验证。

3 访问时间感知的轨迹自监督学习

本章旨在构建挖掘访问时间信息的时空轨迹自监督学习方法，提出一种时间感知的地点嵌入模型 TALE。TALE 包含一种新颖的时间哈夫曼树结构，将轨迹点按照访问时间分配到时间节点中，从而提取地点的功能特征。TALE 还包含一种时间软划分机制，以减少信息丢失，并提高访问时间信息建模的准确性。TALE 模型的自监督学习有效性在四个轨迹数据集和三种下游任务上验证。

3.1 本章引言

如第 1.2.1 和 1.2.1 节所述，时空轨迹中的绝对访问时间信息和上下文信息能够体现地点的功能特征。这种特征有利于众多和地点相关的下游任务，包括建模用户的移动行为[1,4,50]、为用户预测或推荐地点[94–96]、人流量预测[97]、对地点或区域的功能进行分类[40,44] 等。

为了构建从轨迹建模地点功能特征的自监督学习模型，现有方法基于第 2.3.2 节中介绍的 word2vec[38,98] 等分布式词向量模型，从轨迹的上下文信息中提取地点功能特征。然而，当应用于轨迹数据时，word2vec 无法考虑绝对访问时间信息。

为了在自监督学习中引入轨迹的绝对访问时间信息，本章提出了一个新颖的**时间感知的地点嵌入**（*Time-Aware Location Embedding*, **TALE**）模型。该模型在 word2vec 模型的基础上，进一步挖掘轨迹中的绝对访问时间信息，能够为地点学习更全面的嵌入向量。

从技术角度，TALE 通过时空轨迹的自监督学习，得到了一个输入对象为第 2.1.2 节介绍的地点的嵌入映射模型。此类模型形式化定义如下。

定义 3.1 (地点嵌入映射模型，Location Embedding Model)**.** 一个地点嵌入映射模型 f_θ 对于给定的地点 l，将其映射为对应的嵌入向量 $\boldsymbol{z}_l \in \mathbb{R}^d$，即 $f_\theta(l) = \boldsymbol{z}_l$，其中维度 d 被视为超参数。该嵌入向量包含关于地点的潜在信息，例如，地点的功能和地理地点，与其他地点的关系等等。

本研究内容的主要工作总结如下：

1）面向访问时间感知的轨迹自监督学习，提出了时间感知的地点嵌入模型 TALE。TALE 能够挖掘轨迹的访问时间信息，更全面地建模地点功能特征。

2）设计了一种新颖的时间哈夫曼树结构来挖掘轨迹中的访问时间信息。该结构将一天划分为 T 个时间片，对应 T 个时间节点和 T 棵以这些节点为根节点的哈

夫曼子树。每棵子树包含在相应时间片内被访问的地点，并根据地点被访问频率构建。

3）提出了一种时间软划分机制，减少时间片划分造成的访问时间信息丢失，进一步提升地点功能特征挖掘的有效性。

4）所提出的自监督学习模型被应用于三种下游任务，并在四个轨迹集上进行实验。实验结果显示，下游任务的性能得到了显著的提升，证明了所提出模型的有效性。

3.2 时间感知的地点嵌入模型

为了构建能够建模访问时间信息的轨迹自监督学习方法，本节提出时间感知的地点嵌入（Time-Aware Location Embedding, TALE）模型。TALE 基于第 2.3.2 章中介绍的连续词袋模型（Continuous Bag-of-Words, CBOW）[38] 框架搭建地点嵌入模型。

3.2.1 基础地点嵌入模型

为了从轨迹中提取地点的特征信息，CBOW 旨在让具有相似上下文环境的地点的嵌入向量相似，可用于挖掘第 1.2.1 节中介绍的上下文信息。具体来说，给定一条轨迹 $\mathcal{T}=\langle(l_1,t_1),(l_2,t_2),\ldots,\ldots,(l_{|\mathcal{T}|},t_{|\mathcal{T}|})\rangle$ 中的某一个地点 l_i，定义 $C(l_i)=\{l_j\in\mathcal{T},l_j\neq l_i,|j-i|\leq\varepsilon\}$ 为 $\mathcal{T}$ 中 l_i 的上下文（Context），其中 ε 为控制上下文窗口大小的超参数。在 CBOW 框架中，任意地点 l_i 均由两条向量来表示，其中一条为输入向量 $\boldsymbol{z}_{l_i}$，另一条为输出向量 $\boldsymbol{z}'_{l_i}$。输入向量被视为 l_i 的嵌入向量，而输出向量仅用于训练过程。CBOW 的训练目标是给定 $C(l_i)$ 的情况下，最大化目标地点 l_i 出现的概率。该概率计算为：

$$P(l_i|C(l_i))=\exp({\boldsymbol{z}'_{l_i}}^{\top}\phi(C(l_i)))/Z(C(l_i)), \tag{3-1}$$

其中，$\phi(C(l_i))=\sum_{l_j\in C(l_i)}\boldsymbol{z}_{l_j}$，$Z(C(l_i))=\sum_{l_k\in\mathbb{L}}\exp(\boldsymbol{z}_{l_k}^{\top}\phi(C(l_i)))$ 是归一化因子。CBOW 的训练目标即为对 $\mathcal{T}$ 中的所有目标地点-上下文对最大化上述概率。

计算公式 (3-1) 中的 $Z(C(l_i))$ 时，需要遍历整个地点集合 $\mathbb{L}$，因此较为耗时。分层 Softmax 技术可以有效缩短其计算时间。实现这一技术需要将每个地点初始化为一个叶节点，并基于 $\mathbb{T}$ 中的地点出现的频率来构建哈夫曼树（Haffuman Tree）。该树的结构如图 3-1 所示。其中，每个内部节点 v_i 可被视为一个二分类器，包含

隐藏参数向量 $\boldsymbol{\Psi}_{v_i}$。概率 $P(l_i|C(l_i))$ 可计算为从根节点 v_0 到叶节点 l_i 的路径概率：

$$P(l_i|\phi(C(l_i))) = \prod_{v_j \in \mathbb{P}_{v_0,l_i}} \sigma(\mathbb{1}_{v_j} \cdot \boldsymbol{\Psi}_{v_j}^{\top} \phi(C(l_i))), \tag{3-2}$$

其中，$\sigma(x) = 1/(1+e^{-x})$ 是 Sigmoid 函数，$\mathbb{P}_{v_0,l_i}$ 表示从 v_0 到 l_i 的路径中出现的内部节点集合，$\mathbb{1}_{v_j}$ 是一个特殊函数，定义为：

$$\mathbb{1}_{v_j} = \begin{cases} -1 & \text{如果 } v_j \text{ 在路径中选择了左子节点} \\ 1 & \text{其他情况} \end{cases}, \tag{3-3}$$

换言之，公式 (3-2) 中的 $\sigma(\mathbb{1}_{v_j} \cdot \boldsymbol{\Psi}_{v_j}^{\top} \phi(C(l_i)))$ 可以被视为内部节点 v_j 的二分类结果。

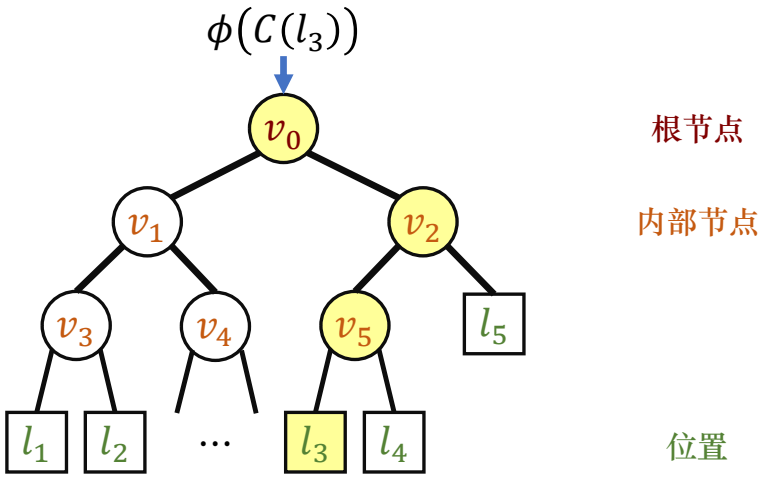


图 3-1　根据轨迹构建的哈夫曼树结构

Figure 3-1 Huffman tree structure constructed from trajectories.

易证 $\sum_{l_i \in L} P(l_i|\phi(C(l_i))) = 1$，这意味着分层 Softmax 计算得到的概率是有效的多项分布。相较于公式 (3-1) 的 $O(L)$ 的时间复杂度，使用哈夫曼树结构能够在 CBOW 的整个训练过程中实现最短的平均路径长度，并将时间复杂度降低到 $O(\log|L|)$。

3.2.2　时间哈夫曼树结构

如第 1.2.1 节所述，地点的功能特征与其被访问时间之间存在强烈的关联性。例如，如果一个用户在工作日的早上 9 点抵达某个地方，那么这个地方很可能是工作场所。如果用户在工作日的晚上 6 点抵达某地，那么这个地方可能是交通枢纽或餐馆。图 1-4 的多条轨迹中，地点 l_i 被三位用户分别在 t_1，t_2 和 t_4 时访问，这意味着 l_1 可能是一个多功能地点。同时，地点 l_2 只在 t_3 时被用户访问，这意味着

l_2 可能是一个单功能地点。另外，地点 l_2 和 l_3 在同一时间 t_3 被用户访问，这意味着它们可能具有相似的功能。显然，将上述访问时间中蕴含的信息融入地点嵌入向量可以提升嵌入的全面性。

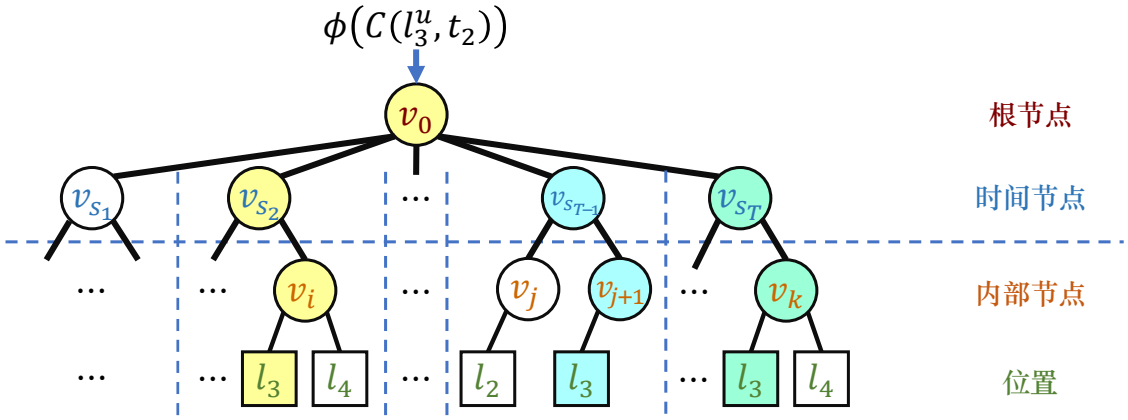


图 3-2　TALE 模型的时间哈夫曼树结构

Figure 3-2 The Temporal Huffman Tree structure of the TALE model.

算法 3.1 : 时间哈夫曼树结构的构建

输入: 地点集合 $\mathbb{L}$，历史轨迹集合 $\mathbb{T}$，时间片长度 ι_{slice}

输出: 时间哈夫曼树结构

初始化 $T = \lceil 24/\iota_{\text{slice}} \rceil$ 和时间片集合 $\{s_1, s_2, \ldots, s_T\}$，其中时间片 s_τ 对应时间跨度 $[\tau \cdot \iota_{\text{slice}}, (\tau+1) \cdot \iota_{\text{slice}}]$;

创建时间节点集合 $\mathbb{V} = \{v_{s_1}, v_{s_2}, \ldots, v_{s_T}\}$，其中时间节点 v_{s_τ} 对应时间片 s_τ;

创建一个根节点 v_0 并将所有时间节点设置为其子节点;

for 每个时间节点 $v_{t_\tau} \in \mathbb{V}$ **do**

  收集访问记录集 $\mathbb{H}_\tau = \{(l_i, t_i) | (l_i, t_i) \in \mathcal{T}, \mathcal{T} \in \mathbb{T}, t_i \in [\tau \cdot \iota_{\text{slice}}, (\tau+1) \cdot \iota_{\text{slice}}]\}$;

  收集地点集 $\mathbb{L}_\tau = \{l_i | l_i$ 在 $\mathbb{H}_\tau$ 中出现 $\}$ 并计算 $\mathbb{H}_\tau$ 中每个地点 $l_i \in L_\tau$ 的出现频率;

  根据地点出现频率构建哈夫曼子树 $\mathcal{H}_\tau$;

  将时间节点 v_{s_τ} 设置为树 $\mathcal{H}_\tau$ 的根节点;

end

返回根节点 v_0，时间节点集合 $\mathbb{V}$ 和哈夫曼子树集合 $\{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_T\}$ 作为时间哈夫曼树结构;

为了将轨迹中的时间信息纳入地点嵌入，TALE 提出了一种新颖的用于 Softmax 计算的时间哈夫曼树（Temporal Huffman Tree）结构。该结构自上而下包含两个部分，如图 3-2 所示。其中，上半部分是一棵两层的多分支树，第一层只包含根

节点 v_0，第二层包含 T 个时间节点 $\{v_{s_1},\ldots,v_{s_T}\}$。将一天的时间均匀地划分为 T 个等长的时间片 $\{s_1,s_2,\ldots,s_T\}$，每个时间片 s_τ 对应一个时间节点 v_{s_τ}。每个时间片的长度被视为一个超参数，记作 ι_{slice}。时间哈夫曼树的下半部分为多棵哈夫曼子树，由轨迹数据集构建。具体来说，$\mathbb{T}$ 中的地点根据被访问时间被分配到对应的时间片中，而时间片对应的时间节点则为此时间片内地点被访问频率对应的哈夫曼子树的根节点。时间哈夫曼树的完整构造过程在算法 3.1 中给出。

3.2.3 时间软划分机制

由于地点通常会在一天的不同时间段被访问，因此一个地点很可能会被分配到多个时间片中。然而，时间片的硬划分与时间的连续性是有冲突的，且将导致一定程度的时间信息丢失。假设有两个餐馆分别在上午 11:55 和下午 12:05 被访问。虽然它们是同种功能的地点，且被访问时间非常接近，但如果将一天划分为 24 个时间片，这两个地点将会被分配到不同的时间片中，使得地点嵌入无法建模它们之间的时间相关性。

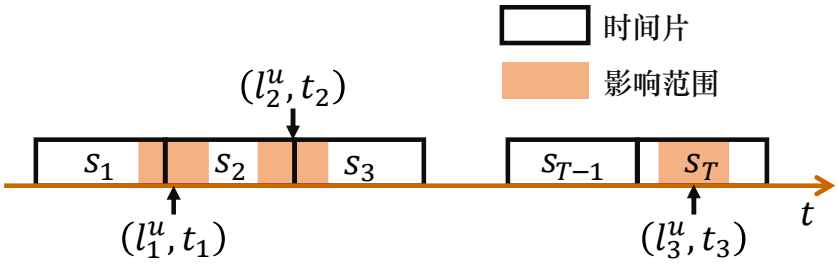


图 3-3　影响范围与多个时间片重叠的示意图

Figure 3-3 The illustration of influence span overlapping multiple time slices.

为了防止时间信息的丢失，TALE 具备一种特殊的时间软划分机制，将轨迹中的一条访问记录 (l_i,t_i) 分配到多个时间片段，使得记录的时间影响可以进一步扩散。具体来说，定义访问记录 (l_i,t_i) 的影响跨度为一个以 t_i 为中心，长度为 ι_{influ} 的时间段，即 $[t-\iota_{\text{influ}}/2,t+\iota_{\text{influ}}/2]$。如果 (l_i,t_i) 的影响跨度与时间片 s_τ 重叠，即 $[t-\iota_{\text{influ}}/2,t+\iota_{\text{influ}}/2]\cap s_\tau\neq\varnothing$，这条记录就会被分配到 s_τ 中。以图 3-3 所示的 T 个时间片为例，地点 l_2 在时间 t_2 被访问，且时间 t_2 的影响跨度与时间片段 s_2 和 s_3 均有重叠，因此该记录同时被分配到 s_2 和 s_3。在时间软划分机制下，构建时间哈夫曼树的过程大体上与算法 3.1 中描述的相同，只是在步骤 5 中，记录集 $\mathbb{H}_\tau$ 的定义

修改为：

$$\begin{aligned}\mathbb{H}_\tau = \{(l_i,t_i)|(l_i,t_i) \in \mathcal{T}, \mathcal{T} \in \mathbb{T},\\ [t - \iota_{\text{influ}}/2, t + \iota_{\text{influ}}/2] \cap\\ [\tau \cdot \iota_{\text{slice}}, (\tau+1) \cdot \iota_{\text{slice}}] \neq \varnothing\}\end{aligned} \tag{3-4}$$

一条访问记录最多会被分配到 $\lceil \iota_{\text{influ}}/\iota_{\text{slice}} \rceil + 1$ 个时间片。定义访问记录 (l_i,t_i) 被分配到的时间片的集合为 $\Omega^{(l_i,t_i)}$。如果访问记录 (l_i,t_i) 被分配到了多个的时间片，计算该记录属于时间片 s_τ 的概率为：

$$P(s_\tau) = \mathcal{L}_{s_\tau}^{(l_i,t_i)} / \sum_{s_\kappa \in \Omega^{(l_i,t_i)}} \mathcal{L}_{s_\kappa}^{(l_i,t_i)}, \tag{3-5}$$

其中，$\mathcal{L}_{s_\tau}^{(l_i,t_i)}$ 是影响跨度 $[t - \iota_{\text{influ}}/2, t + \iota_{\text{influ}}/2]$ 和时间片 s_τ 之间重叠的长度。如图 3-2 所示，一个地点在时间哈夫曼树中可能有多条路径，每条路径都属于一个不同的时间片。例如，l_3 分别在时间片 s_2，s_{T-1} 和 s_T 中出现三次。TALE 能够学习并融合来自不同时间片的地点的功能特征。

综合时间哈夫曼树结构和软划分机制，TALE 与原始的 CBOW 相比有两方面的优势。首先，TALE 将轨迹点的时间影响纳入到 Softmax 分层树结构的构造过程中。出现在相同时间片内的地点往往展现出相似的功能性，因此 TALE 可以将更丰富的功能信息融入到地点嵌入中。其次，在 CBOW 构造的哈夫曼树中，每个地点仅出现一次；而在 TALE 的时间哈夫曼树中，一个地点可能出现多次，使得 TALE 能够更全面地学习多功能地点的嵌入。

3.2.4 概率估计

基于 TALE 的时间哈夫曼树结构，给定一条轨迹 $\mathcal{T}$、目标记录 $(l,t) \in \mathcal{T}$ 和上下文 $C(l,t)$，可以使用层次化 Softmax 方法计算目标记录出现的概率 $P(l,t|C(l,t))$。

具体来说，时间哈夫曼树中的叶节点对应地点，其他节点为内部节点。根节点 v_0 有 T 个分支，可以视为一个多类分类器。其他内部节点可以视为二分类器，和原始的哈夫曼树类似。从根节点 v_0 到时间片 s_τ 中的叶节点 l 的路径可以被定义为一条节点序列，记为 $\mathbb{P}_{v_0,l} = (v_0^{s_\tau}, v_{s_\tau}^l, v_1^l, v_2^l, \ldots, v_n^l)$。由于时间哈夫曼树分为上下两部分，$\mathbb{P}_{v_0,l}$ 也可被对应地划分为两段，即 $\mathbb{P}_{v_0,l} = \mathbb{P}_{v_0,l}^{(1)}, \mathbb{P}_{v_0,l}^{(2)}$。第一段，$P_{v_0,l}^{(1)} = (v_0^{s_\tau})$，只包含选择时间片 s_τ 的分支的根节点。第二段，$\mathbb{P}_{v_0,l}^{(2)} = (v_{s_\tau}^l, v_1^l, v_2^l, \ldots, v_n^l)$ 中的节点均属于一棵哈夫曼子树 $\mathcal{H}_\tau$，时间节点 v_{s_τ} 为该子树的根节点。沿着 $\mathbb{P}_{v_0,l}$ 观察到 l 落

在时间片 s_τ 中的概率计算为：

$$\begin{aligned} P(l_i, s_\tau | C(l,t)) &= P(v_0^{s_\tau} | \phi(C(l,t))) \prod_{v_j^l \in \mathbb{P}_{v_0,l}^{(2)}} P(v_i^l | \phi(C(l,t))) \\ &= P(s_\tau | C(l,t)) \cdot P(l | C(l,t), s_\tau), \end{aligned} \tag{3-6}$$

换言之，给定上下文 $C(l,t)$，地点 l 在时间片 s_τ 内被访问的概率是两个部分的乘积：到达时间在 s_τ 内的概率，以及在 s_τ 时间段内访问的地点是 l 的概率。对于分配到多个时间片 $\Omega^{(l,t)}$ 的访问记录，可以根据公式 (3-5) 计算 l 在时刻 t 被访问的概率，方法是将所有可能路径的概率相加：

$$P(l,t | C(l,t)) = \sum_{s_\tau \in \Omega^{(l,t)}} P(s_\tau) \cdot P(l, s_\tau | C(l,t)) \tag{3-7}$$

接下来详细计算公式 (3-6) 中两个部分的概率。时间哈夫曼树的根节点 v_0 包含一个潜在参数矩阵 $\boldsymbol{M} \in \mathbb{R}^{T \times d}$，可以视为多分类器的参数。如此，公式 (3-6) 第一部分的计算实际上是一个多分类过程：

$$P(s_\tau | C(l,t)) = \exp(\boldsymbol{M}(\tau)^\top \phi(C(l,t))) / Z(C(l,t)), \tag{3-8}$$

其中，$\boldsymbol{M}(\tau)$ 是 $\boldsymbol{M}$ 的第 τ 行，$\phi(C(l,t)) = \sum_{l_j \in C(l,t)} \boldsymbol{z}_{l_j}$ 是 $C(l,t)$ 中所有地点的输入向量特征之和，$Z(C(l,t)) = \sum_{\kappa=1}^{T} \exp(\boldsymbol{M}(\kappa)^\top \phi(C(l,t)))$ 为归一化因子。

每个内部节点 v_{s_τ} 和 v_i $(i \geq 1)$ 包含一个潜在参数向量 $\boldsymbol{\Psi}_{v_i} \in \mathbb{R}^d$，可以视为二分类器的参数。如此，路径 $\mathbb{P}_{v_0,l}^{(2)}$ 中的每个节点均可以看作进行了一个二分类过程，形式化为：

$$P(v_i^l | \phi(C(l,t))) = \sigma(\mathbb{1}_{v_i^l} \cdot \boldsymbol{\Psi}(v_i^l)^\top \phi(C(l,t))), \tag{3-9}$$

其中，$\sigma(x) = 1/(1+e^{-x})$ 是 Sigmoid 函数，$\mathbb{1}_{v_i^l}$ 的定义参见公式 (3-3)。公式 (3-6) 中的第二段定义为路径 $\mathbb{P}_{v_0,l}^{(2)}$ 中节点的联合概率：

$$P(l | C(l,t), s_\tau) = \prod_{v_i^l \in \mathbb{P}_{v_0,l}^{(2)}} P(v_i^l | \phi(C(l,t))) \tag{3-10}$$

最后，以图 3-2 中的访问记录 (l_3, t_2) 为例，假设 t_2 在时间片 s_2 内。地点 l_3 在时间片 s_2 中的路径为 $\mathbb{P}_{v_0,l_3} = (v_0^{s_2}, v_{s_2}^{l_3}, v_i^{l_3})$。这条路径的概率为：

$$\begin{aligned} &P(l_3, s_2 | C(l_3,t_2)) \\ &= \frac{e^{(\boldsymbol{M}(2)^\top \phi(C(l_3,t_2)))}}{Z(C(l_3,t_2))} \times \sigma(\boldsymbol{\Psi}_{v_{s_2}}^\top \phi(C(l_3,t_2))) \times \sigma(-\boldsymbol{\Psi}_{v_i}^\top \phi(C(l_3,t_2))) \end{aligned} \tag{3-11}$$

在 TALE 的时间哈夫曼树结构中，叶节点的最大数量是 $(T \times |\mathbb{L}|)$，其中 T 是时间切片的数量。所有叶节点的平均路径长度是 $log|\mathbb{L}| + 1$。因此，公式 (3-6) 的

时间复杂度为 $O(\log|\mathbb{L}|+1)$，比公式 (3-1) 的时间复杂度 $O(|\mathbb{L}|)$ 低得多。换言之，TALE 在建模轨迹中时间信息的基础上，依然能相较原版 CBOW 大幅提升计算效率。

3.2.5 模型训练

TALE 模型的训练目标是在给定轨迹中目标访问记录的上下文的情况下，最大化观察到目标访问记录的后验概率。假设数据集中的轨迹彼此独立，那么模型训练目标可形式化为：

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{argmax} \prod_{(l,t)\in\mathcal{T},\ \mathcal{T}\in\mathbb{T}} P(l,t|C(l,t)), \tag{3-12}$$

其中 $\boldsymbol{\Theta}=\{\boldsymbol{Z},\boldsymbol{Z}',\boldsymbol{M},\boldsymbol{\Psi}\}$ 为 TALE 的所有可学习参数，$\boldsymbol{Z}$ 是地点输入向量的集合，$\boldsymbol{Z}'$ 是地点输出向量的集合，$\boldsymbol{M}$ 是时间哈夫曼树结构中根节点的参数，$\boldsymbol{\Psi}$ 是所有其他内部节点的参数的集合。随机梯度下降（Stochastic Gradient Descent, SGD）方法[99] 被用于进行模型参数训练。

最后，收敛得到的最优参数 $\boldsymbol{\Theta}^*$ 中的地点输入向量的集合 $\boldsymbol{Z}^*$ 即为学习到的地点嵌入向量的集合。对于每个地点 l_i，$\boldsymbol{Z}^*$ 的第 l_i 行对应地点的嵌入向量。

3.3 实验

为了展示 TALE 模型学习到的地点嵌入在轨迹数据挖掘中的有效性，本节将其纳入多种下游应用，并在四个真实世界的时空轨迹数据集上与其他嵌入方法的效果对比。

3.3.1 基线模型

为了证明 TALE 模型学习到的嵌入向量的有效性，本节选取了一些经典的词嵌入模型和最先进的地点嵌入模型与其进行比较。

- **CBOW**[38]：word2vec[98] 的一种实现方案，在第 3.2.1 节中详细介绍。
- **Skip-Gram**[38]：word2vec 的另一种实现方案。不同于 CBOW，它将目标词作为输入，上下文词作为输出。
- **POI2Vec**[43]：基于 CBOW 的地点嵌入模型。该模型认为地点之间的地理关系会影响用户的移动行为，并在嵌入学习过程中融入了地理特征。

• **Geo-Teaser**[42]：基于 Skip-Gram 的地点嵌入模型。通过将时间状态向量与目标地点的嵌入向量连接作为 Skip-Gram 的输入，将工作日或周末的影响纳入到地点嵌入中。

3.3.2 实验设置

TALE 和各基线模型在第 2.4.3 节中介绍的地点分类、访问流量预测和轨迹预测任务上验证。具体而言，地点分类的适配方案（1）、（2）分别由全连接网络（**FC**）和 ***k*NN** 算法[81] 实现。轨迹预测采用方案（1）实现，根据历史轨迹预测未来一个点的 POI 地点下标，序列预测模型选用 **GRU**[100] 和 **DeepMove**[101]。

考虑到 TALE 对地点功能特征的建模要求轨迹中的地点均有明确的访问语意信息，实验选用第 2.2.1 节中介绍的 Foursquare-NYC、Foursquare-TKY、Foursquare-JKT 签到记录数据集以及第 2.2.2 节中介绍的 Mobile-PEK 手机信令数据集。对于给定的轨迹数据集，首先计算所有访问记录的最早和最晚时间戳，然后沿时间轴按照 6:2:2 的比例将轨迹划分为训练、评估和测试轨迹集。地点嵌入模型只在训练轨迹集上进行训练。在地点分类任务中，地点集也以 6:2:2 的比例被划分为训练、评估和测试地点集。所有下游预测模型在训练集上进行训练，在评估集上实现早停技术，并在测试集上计算最终结果。需要说明的是，地点的类标签只在 Foursquare 签到数据集中可用，因此本实验只在这些数据集上进行地点分类任务；Foursquare 签到数据集的访问记录太稀疏，难以计算访问流量，因此本实验只在 Mobile-PEK 数据集上进行访问流量预测任务。

TALE 模型基于基于 PyTorch 框架[102] 实现。在实验中，TALE 模型的嵌入向量的维度 d 设为 128，上下文窗口大小 ε 设为 2，时间片长度 ι_{slice} 设定为 240 分钟，影响跨度长度 ι_{influ} 设定为 60 分钟。所有模型的批次大小设为 64，随机梯度下降优化器的初始学习率为 0.001。访问流量预测模型使用均方误差损失进行训练，地点分类模型和轨迹预测模型使用交叉熵损失进行训练。

3.3.3 总体性能对比

表 3-1 至 3-3 分别展示了不同地点嵌入方法在地点分类，访问流量预测和轨迹预测任务中的性能比较。TALE 在大部分指标上均明显优于基线方法。

CBOW 和 Skip-Gram 直接从自然语言处理领域迁移过来，忽略了轨迹中独特的空间和时间信息。当这两种方法生成的地点嵌入向量应用于下游模型时，它们的表现通常是最差的。相比之下，POI2Vec 基于用户倾向于访问附近地点的这一理

念，在地点嵌入中考虑了空间影响。然而，在地点嵌入中认为地点的空间相近性和功能相关性有直接联系并不全面和准确，因为在现实世界中，小范围内不同地点的功能性也可能会有很大差异。

时间信息与地点的功能特征有很强的相关性，因为具有相近功能的地点会有相似的访问时间分布。Geo-Teaser 将时间影响和空间信息一同纳入模型。但对于时间影响，它只区分了工作日和周末发生的地点访问记录，因此无法挖掘更细粒度的时间信息。与上述基线相比，TALE 使用了时间哈夫曼树结构，模拟相似时间段内被访问的地点之间的关联性，从而将更详细、准确的地点功能信息融入学习到的地点嵌入向量中。因此，下游预测任务与 TALE 结合时，达成了最高的准确率。

表 3-1　不同方法在访问流量预测任务上的性能比较

Table 3-1 Performance comparison of different approaches towards visiting flow prediction.

指标 嵌入方法	MAE	RMSE
Skip-gram	2.835±0.01	4.258±0.03
CBOW	2.771±0.01	4.143±0.02
POI2Vec	2.599±0.01	3.871±0.03
Geo-Teaser	<u>2.535±0.01</u>	<u>3.773±0.02</u>
TALE	**2.399±0.01**	**3.560±0.02**

粗体表示最佳结果，<u>下划线</u> 表示次好结果。

3.3.4　模型分析

(1) 超参数效果分析

本节评估三个超参数的效果：时间片长度 ι_{slice}、影响范围长度 ι_{influ} 和嵌入向量维度 d。实验在基于 DeepMove 模型的轨迹预测任务上进行。在评估某一超参数时，其他参数锁定为最优。

时间片长度 ι_{slice}： 图 3-4(a) 显示了时间片长度 ι_{slice} 对预测准确率的影响。可以观察到，随着时间片的延长，预测准确率首先上涨，超过最优点后开始下跌。过小的 ι_{slice} 意味着只有在非常接近的时间间隔内被访问的地点才会被划分到同一时间片中，导致相似时间段内访问的地点之间的关系无法被捕获。过大的 ι_{slice} 将太多的地点划分到同一个时间片中，无法精细地建模轨迹中的时间信息，甚至退化

表 3-2　不同方法在地点分类任务上的性能比较

Table 3-2 Performance comparison of different approaches towards location classification.

预测模型		FC					*k*NN	
指标		Acc@1 (%)	Acc@5 (%)	Acc@10 (%)	Acc@20 (%)	macro-F1 (%)	Acc@1 (%)	macro-F1 (%)
数据集	嵌入方法							
Foursquare-NYC	Skip-gram	18.465±0.19	33.453±0.47	44.220±0.52	58.798±0.82	1.626±0.15	6.266±0.72	1.811±0.22
	CBOW	18.414±0.36	33.542±0.57	43.811±0.33	57.583±0.64	1.651±0.18	6.532±0.89	1.694±0.27
	POI2Vec	19.587±0.34	36.019±0.46	47.187±0.66	61.679±0.64	1.954±0.26	7.015±0.92	1.749±0.24
	Geo-Teaser	21.723±0.81	39.290±0.62	49.457±0.93	62.852±0.79	2.606±0.38	7.399±0.33	1.918±0.33
	TALE	**22.232±0.43**	**41.176±1.11**	**51.005±0.75**	**64.194±0.55**	**2.664±0.35**	**7.709±0.62**	**2.228±0.39**
Foursquare-TKY	Skip-gram	17.783±0.55	39.542±0.65	53.864±0.45	68.288±0.60	4.196±0.42	13.031±0.56	2.607±0.24
	CBOW	17.057±0.38	39.324±0.55	53.432±0.96	67.601±0.95	3.855±0.52	12.535±0.57	3.595±0.17
	POI2Vec	18.944±0.56	40.667±0.89	54.171±0.70	68.670±0.51	4.584±0.61	13.414±0.65	3.556±0.41
	Geo-Teaser	19.283±0.36	41.053±0.43	55.043±0.84	69.795±0.89	4.547±0.35	14.164±0.17	3.086±0.19
	TALE	**20.516±0.52**	**42.408±0.85**	**55.645±1.02**	**70.534±0.66**	**4.793±0.66**	**15.492±0.66**	**4.090±0.31**
Foursquare-JKT	Skip-gram	5.914±0.39	19.210±0.29	30.485±0.53	45.345±0.94	1.106±0.07	2.724±0.15	0.955±0.14
	CBOW	5.779±0.28	19.233±0.50	31.251±0.59	46.578±0.56	1.251±0.12	2.772±0.10	1.042±0.17
	POI2Vec	6.620±0.41	20.282±0.66	32.125±0.60	47.222±0.48	1.456±0.43	2.943±0.26	1.123±0.13
	Geo-Teaser	6.725±0.32	21.302±0.76	33.216±0.84	**48.595±0.84**	**1.666±0.23**	2.976±0.26	1.009±0.21
	TALE	**7.044±0.37**	**21.576±0.23**	**33.766±0.65**	48.522±0.27	1.578±0.39	**3.256±0.32**	**1.198±0.21**

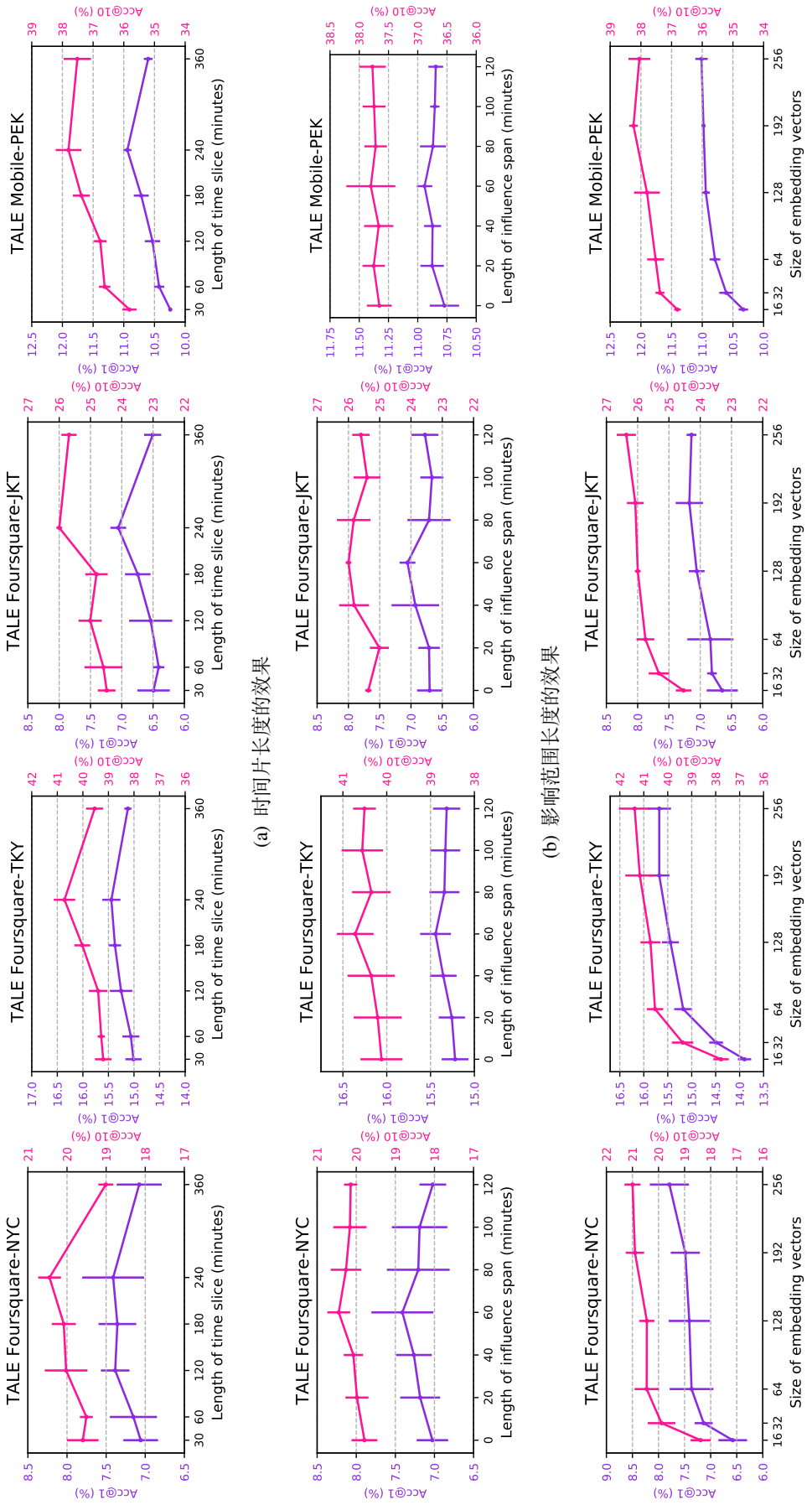
粗体表示最佳结果，下划线 表示次好结果。

表 3-3　不同方法在轨迹预测任务上的性能比较

Table 3-3 Performance comparison of different approaches towards trajectory prediction.

预测模型		GRU					DeepMove				
指标		Acc@1 (%)	Acc@5 (%)	Acc@10 (%)	Acc@20 (%)	macro-F1 (%)	Acc@1 (%)	Acc@5 (%)	Acc@10 (%)	Acc@20 (%)	macro-F1 (%)
数据集	嵌入方法										
Foursquare-NYC	Skip-gram	5.466±0.23	11.870±0.33	14.986±0.34	18.878±0.37	1.254±0.12	6.170±0.21	13.351±0.52	16.555±0.58	20.310±0.56	1.354±0.16
	CBOW	5.248±0.22	10.849±0.22	13.522±0.16	16.367±0.30	1.203±0.09	6.191±0.10	13.282±0.47	16.304±0.49	19.814±0.40	1.556±0.11
	POI2Vec	5.935±0.19	12.529±0.35	15.910±0.30	19.961±0.31	1.471±0.10	6.357±0.28	13.786±0.25	17.239±0.22	21.237±0.31	1.515±0.12
	Geo-Teaser	6.209±0.32	13.677±0.43	17.602±0.37	21.996±0.39	1.533±0.11	7.007±0.28	15.472±0.41	19.458±0.17	23.741±0.56	1.717±0.08
	TALE	**6.918±0.39**	**14.874±0.44**	**18.911±0.34**	**23.018±0.31**	**1.685±0.20**	**7.410±0.40**	**16.455±0.41**	**20.447±0.29**	**24.711±0.19**	**1.867±0.10**
Foursquare-TKY	Skip-gram	10.705±0.17	25.417±0.27	32.986±0.30	40.944±0.34	1.650±0.08	13.173±0.14	28.965±0.23	36.651±0.28	44.529±0.32	2.269±0.07
	CBOW	10.373±0.10	23.899±0.21	30.690±0.24	37.899±0.23	1.720±0.09	12.823±0.18	27.391±0.42	34.340±0.43	41.528±0.39	2.350±0.15
	POI2Vec	11.425±0.13	26.785±0.22	34.570±0.26	42.854±0.26	2.023±0.04	14.306±0.11	30.532±0.21	38.533±0.17	46.665±0.15	2.958±0.01
	Geo-Teaser	12.024±0.18	28.696±0.38	37.176±0.40	45.851±0.45	2.413±0.10	14.669±0.15	32.231±0.28	**41.989±0.44**	**50.540±0.60**	3.150±0.10
	TALE	**12.637±0.16**	**29.753±0.39**	**38.476±0.40**	**47.239±0.37**	**2.848±0.10**	**15.443±0.18**	**33.453±0.27**	40.720±0.42	49.077±0.47	**3.622±0.07**
Foursquare-JKT	Skip-gram	5.524±0.17	14.100±0.17	19.853±0.20	26.840±0.14	0.776±0.07	5.713±0.15	14.423±0.23	20.498±0.26	27.654±0.15	0.835±0.13
	CBOW	5.290±0.07	13.773±0.25	19.367±0.31	26.055±0.35	0.898±0.09	5.736±0.16	14.309±0.22	19.965±0.27	26.815±0.29	1.025±0.15
	POI2Vec	5.876±0.17	15.356±0.24	21.650±0.27	29.392±0.19	0.846±0.09	6.229±0.11	15.904±0.07	22.350±0.32	30.029±0.43	1.076±0.16
	Geo-Teaser	**6.376±0.13**	16.740±0.30	23.808±0.50	32.148±0.37	1.025±0.12	6.533±0.11	17.225±0.40	24.248±0.31	32.300±0.30	1.444±0.07
	TALE	6.278±0.20	**17.901±0.28**	**25.424±0.44**	**34.132±0.38**	**1.374±0.09**	**7.056±0.13**	**18.640±0.17**	**25.997±0.09**	**34.711±0.31**	**1.469±0.07**
Mobile-PEK	Skip-gram	7.994±0.09	22.574±0.14	31.232±0.18	40.819±0.20	3.453±0.10	8.650±0.10	23.564±0.09	31.703±0.17	40.486±0.22	3.996±0.07
	CBOW	8.458±0.07	22.796±0.16	30.418±0.18	38.517±0.19	4.122±0.05	8.895±0.05	23.980±0.14	31.778±0.16	39.923±0.17	4.632±0.09
	POI2Vec	9.535±0.09	25.761±0.03	34.883±0.14	44.416±0.14	4.789±0.14	9.762±0.15	26.254±0.06	34.977±0.05	44.151±0.16	5.044±0.10
	Geo-Teaser	9.351±0.08	25.680±0.15	34.791±0.17	44.508±0.26	4.523±0.12	9.639±0.06	25.856±0.12	34.517±0.14	43.711±0.21	4.819±0.20
	TALE	**10.939±0.06**	**29.319±0.13**	**38.860±0.19**	**48.490±0.22**	**5.010±0.12**	**10.942±0.07**	**28.479±0.29**	**37.802±0.42**	**47.301±0.54**	**5.524±0.22**

粗体表示最佳结果，下划线 表示次好结果。



(a) 时间片长度的效果　(b) 影响范围长度的效果　(c) 嵌入向量维度的效果

图 3-4　在 DeepMove 预测模型上验证的超参数的影响

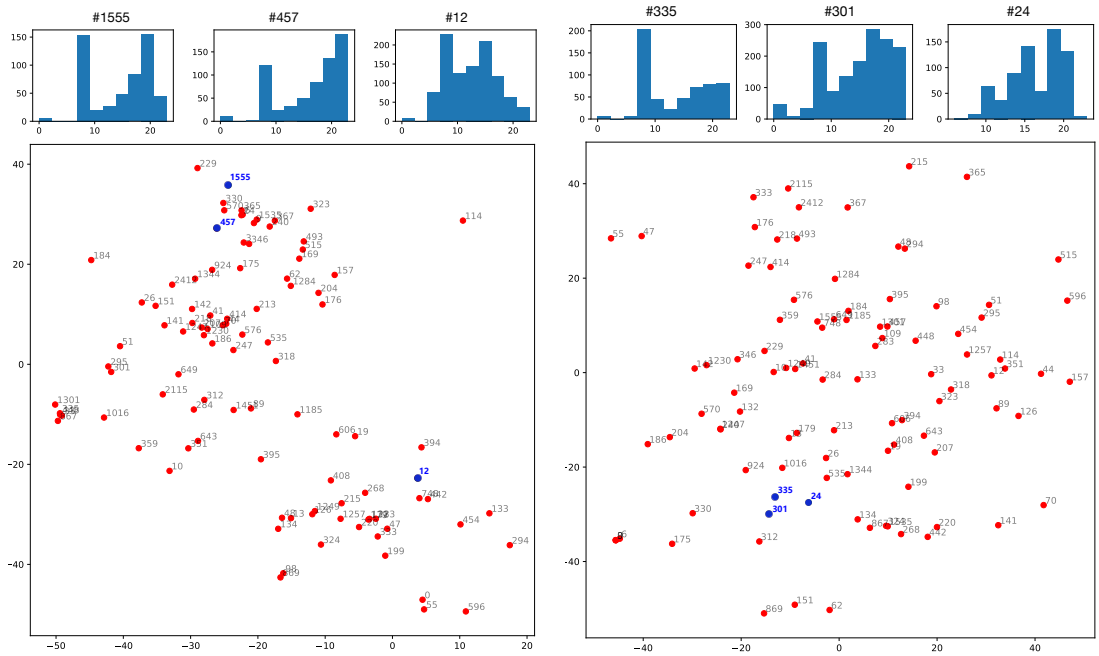
Figure 3-4 Effects of hyper-parameters validated on DeepMove.

为基本的 CBOW[38] 模型。合适大小的 ι_{slice} 可以捕获具有相似访问时间模式的地点之间的关系，同时也可以区分那些在不同时间段内被访问的地点。

影响范围长度 ι_{influ}： 图 3-4(b) 显示了影响跨度长度 ι_{influ} 和预测准确率的关系。过小的 ι_{influ} 会导致在相邻时间段内被访问地点被分配到不同的时间片中，从而丢失这些地点之间的时间关系。过大的 ι_{influ} 会将一个地点分配到无关的时间片中，无法建模地点独特的访问时间模式。

嵌入向量大小 d： 图 3-4(c) 显示了嵌入向量维度 d 的影响。随着维度的增加，预测准确性逐步提升。较长的向量能够包含更全面的信息，从而帮助下游任务获得更好的结果。然而，当维度大于 128 时，扩展为度带来的性能改善有限，同时将导致更高的计算成本。因此，将嵌入维度设置为 128 时达到了效率和效果之间的平衡。

(2) 地点嵌入向量的可视化



(a) TALE 学习到的地点嵌入向量的可视化。 (b) CBOW 学习到的地点嵌入向量的可视化。

图 3-5　不同方法学习到的地点嵌入向量的可视化

Figure 3-5 Visualization of location embedding vectors learned by different methods.

本节选择 Foursquare-TKY 数据集中的少量地点，并将它们的嵌入空间映射到二维空间中进行可视化，以直观地观察它们在嵌入空间中的关系。使用 t-SNE 方法[103] 对地点的嵌入向量进行降维后，可以在可视化图中观察到这些地点嵌入向

量间的距离；作为参考，这些地点被访问的时间模式被可视化为访问时间直方图。TALE 模型的可视化结果如图 3-5(a) 所示。可以看到，嵌入向量接近的地点通常具备相似的访问时间模式，嵌入向量远离的地点在访问时间模式上具有较大的差异。例如地点 #1555 和 #457 均为火车站，而地点 #12 是地铁站。

对基线模型 CBOW 学习到的嵌入进行相同的可视化流程，结果如图 3-5(b) 所示。很明显，具有不同访问时间模式的地点在 CBOW 获得的嵌入空间中不具备较强的规律性。这意味着 CBOW 忽略时空轨迹数据中的时间信息，仅使用序列性质对地点的特征进行建模，这将会降低学习到的地点嵌入向量的质量。

3.4 本章小结

本章为了解决自监督学习难以从轨迹数据中挖掘访问时间信息的挑战，进行了访问时间感知的轨迹自监督学习相关研究，并提出了一种时间感知的地点嵌入模型 TALE。该模型能够在自监督学习过程中，将轨迹访问时间信息纳入地点嵌入映射模型，进而得到能输出更全面的地点嵌入向量的自监督学习模型。

具体而言，TALE 提出了一种时间哈夫曼树结构，该结构将一天划分为时间片，每个时间片对应一个以时间节点为根节点的哈夫曼子树。在同一时间片内被访问的地点被划分到同一棵哈夫曼子树，因此 TALE 能够建模地点特征与轨迹访问时间的关联性。同时，为了减少时间片划分过程中的信息丢失，TALE 提出了一种时间软划分机制，进一步提升对轨迹访问时间信息建模的准确性。

TALE 在三个签到记录数据集和一个手机信令数据集上进行验证，其自监督学习得到的模型被适配于三种下游任务，即地点分类、访问流量预测和轨迹预测中。与基线模型的对比表明 TALE 所学习的模型能够提升下游任务的准确性，证明了 TALE 自监督学习的有效性。同时，对地点嵌入向量的可视化实验直观地展示了 TALE 能够将轨迹访问时间信息融入自监督学习中。

4 上下文感知的轨迹自监督学习

本章旨在构建挖掘上下文信息的时空轨迹自监督学习方法，提出一种上下文与时间感知的地点嵌入模型 CTLE。CTLE 是一种考虑轨迹中动态上下文信息的自监督学习方法，能够建模多功能地点的准确功能特征。CTLE 还包含一个时间编码模块和小时掩码自监督目标，同时考虑了轨迹的两方面访问时间信息。CTLE 模型的自监督学习有效性在两个轨迹数据集和轨迹预测下游任务上验证。

4.1 本章引言

本文在第 3 章中介绍了一种受 word2vec 模型[38] 启发，从轨迹的上下文和绝对访问时间信息中挖掘地点功能特征的方法 TALE。TALE 通过将轨迹中的绝对访问时间信息融入自监督学习模型，得到了比基线方法更全面的地点嵌入向量。

然而，正如第 1.2.1 所述，轨迹的相对访问时间差也是揭示地点功能特征的关键信息之一。同时，对于多功能地点，轨迹的动态上下文环境能够更准确地体现其具体的功能特征，这在第 1.2.1 节中进行了详细论述。TALE 及其他基于词嵌入模型构建的自监督学习模型的局限性在于，轨迹数据中的每个地点始终被映射为对应的一条嵌入向量，并没有对轨迹中不同地点之间的动态关联性进行建模。因此，这些方法均无法考虑轨迹的动态上下文信息和相对访问时间差信息。

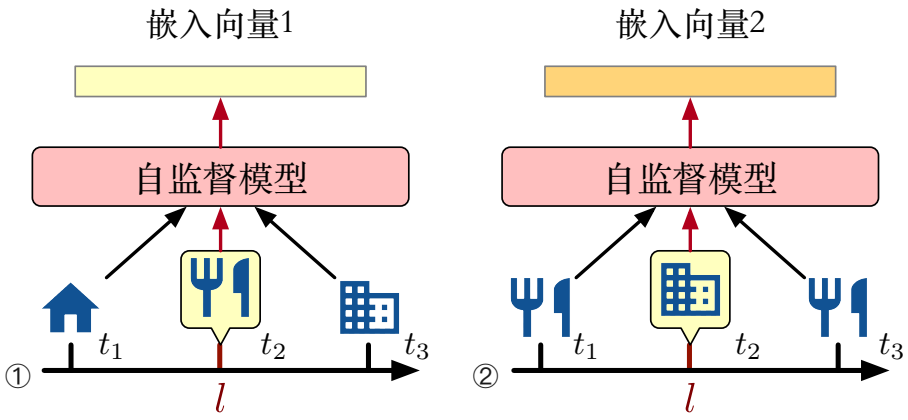


图 4-1 能够动态地将地点映射为嵌入向量的自监督学习模型

Figure 4-1 A self-supervised learning model that can dynamically map locations into embeddings.

为了进一步提升轨迹自监督学习对于地点功能特征建模的全面性和准确性，需要建立一个自监督学习模型，该模型能够根据轨迹的上下文环境和相对访问时

间差，动态地将地点映射为嵌入向量，如图 4-1 所示。为了实现上述任务，本章提出了一种**上下文与时间感知的地点嵌入**（*Context and Time Aware Location Embedding*，**CTLE**）模型。与现有的基于词嵌入的自监督学习方法不同，CTLE 根据轨迹的上下文环境，凭借可学习的映射函数来计算目标地点的嵌入向量。如此，CTLE 能够将地点在不同上下文中的特定功能特征和相对访问时间差信息纳入其输出的嵌入向量中。

从技术角度，CTLE 通过时空轨迹的自监督学习，得到了一个输入对象为地点的嵌入映射模型。和 TALE 类似，其将轨迹中的地点映射为固定维度的嵌入向量。

本研究内容的主要工作总结如下：

1）面向上下文感知的轨迹自监督学习，提出了上下文与时间感知的地点嵌入模型 CTLE。CTLE 能够挖掘轨迹的动态上下文信息和相对访问时间差信息，更准确地建模地点功能特征。

2）提出基于双向 Transformer 和掩码语言模型的轨迹自监督模型，建模轨迹中地点和上下文的关联性，并能生成地点在特定上下文环境下的嵌入向量。

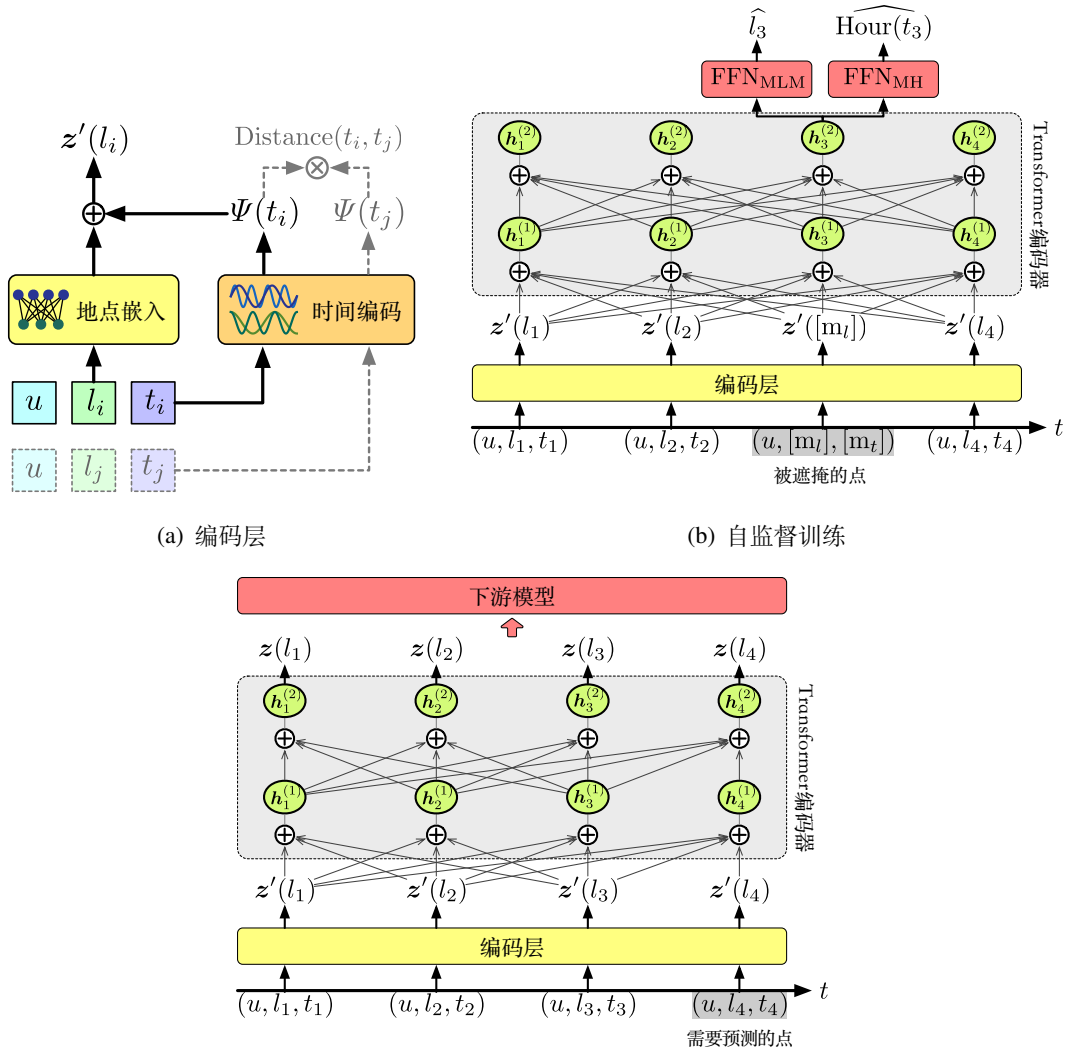
3）提出了一个时间编码模块和掩码小时自监督目标，以建模轨迹中的绝对和相对两方面的访问时间信息，加强模型对地点功能特征的挖掘。

4）所提出的自监督模型与三个下游预测模型结合，以适配至轨迹预测任务，并在两个轨迹数据集上进行了广泛的实验。实验结果显示预测性能显著提高，证明了模型的优越性。

4.2 上下文与时间感知的地点嵌入模型

4.2.1 模型整体结构

为了建模轨迹动态上下文信息，本章提出了上下文感知的地点嵌入（CTLE）自监督模型。图 4-2 展示了 CTLE 模型的架构。它主要由三个组件构成：（1）编码层，该层将时间编码向量与地点嵌入向量融合，使模型能够从相对访问时间差中提取信息；（2）基于双向 Transformer 的编码器，作为计算目标地点嵌入的映射函数；（3）自监督目标，通过建模目标地点与上下文共同出现的概率，以及目标地点被访问的绝对时间，将地点的特征信息整合到它们的嵌入中。



(a) 编码层　　(b) 自监督训练

(c) 计算上下文嵌入

图 4-2　CTLE 模型的架构与构成组件

Figure 4-2 The architecture and components of the CTLE model.

4.2.2　上下文感知的双向序列模型

在现实世界中，多功能地点非常常见。通常，用户对某个地点的访问行为带有特定的目的；但如果该地点是多功能的，那么单条访问记录 (l,t) 并不能准确表示这条记录所处环境下该地点具体的功能。上下文环境，即轨迹中目标地点附近的访问记录，可以更明确地表示目标地点的特定功能。

基于上述观察，本章提出一个考虑上下文环境的地点嵌入模型。具体来说，给定目标地点 l 和其上下文 $C(l)$，该模型通过参数化的映射函数 f 计算地点的嵌入

向量 $\boldsymbol{z}_l$，形式化表示为：

$$\boldsymbol{z}_l = f(l, C(l)) \tag{4-1}$$

这样，地点的嵌入向量与它们的上下文环境相关，使得地点的特定功能能够被区分。双向 Transformer 编码器[73] 被用于实现函数 f。这一选择的原因有两点：首先，轨迹中的地点与左右两侧的上下文都有关，而双向 Transformers 能够从两侧捕捉此关联性；其次，现有研究[104] 证明，Transformer 架构相比传统的序列模型，如 LSTM[72]，具有更强的表达能力。

给定一条轨迹 $\mathcal{T} = \langle (l_1, t_1), (l_2, t_2), \ldots, (l_{|\mathcal{T}|}, t_{|\mathcal{T}|}) \rangle$，首先用地点嵌入模块获取每个地点 l 的输入隐向量 $\boldsymbol{z}'_l$，表示为：

$$\boldsymbol{z}'_l = \Omega(l), \tag{4-2}$$

从而形成一条输入序列 $\langle \boldsymbol{z}'_{l_1}, \boldsymbol{z}'_{l_2}, \ldots, \boldsymbol{z}'_{l_{|\mathcal{T}|}} \rangle$。$\Omega$ 表示地点嵌入模块，使用全连接嵌入层实现。随后，将输入序列作为 Transformer 编码器的输入，其中编码器每层由多头自注意力模块和全连接前馈网络组成。该过程形式化表示为：

$$\begin{aligned} \langle \boldsymbol{h}_1^{(k)}, \boldsymbol{h}_2^{(k)}, \ldots, \boldsymbol{h}_{|\mathcal{T}|}^{(k)} \rangle &= \text{TransEncLayer}(\langle \boldsymbol{h}_1^{(k-1)}, \boldsymbol{h}_2^{(k-1)}, \ldots, \boldsymbol{h}_{|\mathcal{T}|}^{(k-1)} \rangle) \\ \langle \boldsymbol{h}_1^{(0)}, \boldsymbol{h}_2^{(0)}, \ldots, \boldsymbol{h}_{|\mathcal{T}|}^{(0)} \rangle &= \langle \boldsymbol{z}'_{l_1}, \boldsymbol{z}'_{l_2}, \ldots, \boldsymbol{z}'_{l_{|\mathcal{T}|}} \rangle, \end{aligned} \tag{4-3}$$

其中 TransEncLayer 表示 Transformer 编码器层。$\{\boldsymbol{h}_1^{(k)}, \boldsymbol{h}_2^{(k)}, \ldots, \boldsymbol{h}_{|\mathcal{T}|}^{(k)}\}$ 是第 k 层的输出序列，也是第 $(k+1)$ 层的输入序列。假设模型总共包含 N 层 Transformer 编码器层，那么最后一层的输出内存序列的第 i 项，即 $\boldsymbol{h}_i^{(N)}$，视为轨迹 $\mathcal{T}$ 中地点 l_i 的上下文嵌入向量。如果将 $\mathcal{T}$ 中除 l_i 外的所有地点视为其上下文 $C(l_i)$，那么 $\boldsymbol{z}_{l_i}$ 的计算公式可以写为：

$$\begin{aligned} \boldsymbol{z}_{l_i} = \boldsymbol{h}_i^{(N)} &= \text{TransEnc}(\langle \boldsymbol{z}'_{l_1}, \boldsymbol{z}'_{l_2}, \ldots, \boldsymbol{z}'_{l_{|\mathcal{T}|}} \rangle)_i \\ &= \text{TransEnc}(\Omega(\langle l_1, l_2, \ldots, l_n \rangle))_i \\ &= f(\langle l_1, l_2, \ldots, l_n \rangle)_i \\ &= f(l_i, C(l_i))_i, \end{aligned} \tag{4-4}$$

其中，TransEnc 表示 Transformer 编码器，由 Transformer 编码器层堆叠而成。公式 (4-4) 与公式 (4-1) 中给出的抽象形式相对应，因此这种地点嵌入的计算方式能够考虑上下文环境。

通过模型训练，映射函数 f 应当理解目标地点与其对应上下文之间的关系，从而将地点的特性信息融入到其嵌入向量中。受 NLP 领域语言模型的启发，CTLE 实现了第 2.3.2 章中介绍的掩码语言模型（MLM）[29] 来构造自监督目标。给定一条轨迹 $\mathcal{T}$，随机选择 20% 的访问记录替换为特殊的标记 $([\mathrm{m}_l], [\mathrm{m}_t])$，即隐去这些记

录的地点和时间信息。如此得到的轨迹 $\widetilde{\mathcal{T}}$ 作为函数 f 的输入，并借助 f 的输出来预测被掩盖的记录的原始地点：

$$\widehat{l_m} = \text{FFN}_{\text{MLM}}(\boldsymbol{h}_m^{(N)}) = \text{FFN}_{\text{MLM}}(f(\widetilde{\mathcal{T}})_m), \tag{4-5}$$

其中 FFN_{MLM} 为全连接网络，用于将 Transformer 编码器的输出映射为对地点的预测。MLM 训练目标通过最大化预测准确性来构建。如果将被掩盖的地点视为目标，将未被掩盖的地点视为目标的上下文，那么 MLM 训练目标可形式化为：

$$\begin{aligned} O_{\text{MLM}} &= \arg\max_{\theta} \textstyle\sum_{l_m \in \Gamma} P(l_m | \text{FFN}_{\text{MLM}}(f(\widetilde{\mathcal{T}})_m)) \\ &= \arg\max_{\theta} \textstyle\sum_{l_m \in \Gamma} P(l_m | \text{FFN}_{\text{MLM}}(f(C(l_m))_m), \end{aligned} \tag{4-6}$$

其中，Γ 表示轨迹中所有被掩盖的地点的集合，θ 表示 f 的可学习参数集合。通过最大化目标和其上下文的共同出现的概率，具有相似上下文环境的地点将有更接近的嵌入向量。这个想法类似于 word2vec[38] 的训练目标。然而，CTLE 并非直接优化一组静态嵌入向量，而是优化映射函数的参数。如此学习到的嵌入向量将依赖于上下文，能够区分不同上下文环境中地点的特定功能特征。

4.2.3 时间感知的编码层与掩码恢复任务

如本文在第 1.2.1 节中所讨论的，轨迹数据中的时间信息分为两个方面：相对访问时间差和绝对访问时间。为了全面建模两方面的信息，CTLE 中包含一个精致的时间编码模块和一个新颖的自监督目标，将两方面的访问时间信息纳入模型中。

编码层中的时间编码： 为了明确地建模轨迹中的相对访问时间差异，CTLE 的编码层中包含一个时间编码模块。在原始 Transformer[73] 中，位置编码被用于区分输入序列各单元的顺序。其计算形式化为：

$$\boldsymbol{\Phi}(o) = [\cos(\omega_1 o), \sin(\omega_1 o), \ldots, \cos(\omega_d o), \sin(\omega_d o)], \omega_k = 1/10000^{2k/d}, \tag{4-7}$$

其中，函数 $\boldsymbol{\Phi}$ 表示位置编码模块，o 是一个单元的索引，位置编码向量的维度为 $2d$。

对于轨迹数据，地点的访问记录在时间轴上呈非均匀排列，而位置编码仅用于表示等间隔信息。受动态图表示工作[105] 的启发，CTLE 对位置编码进行了两个简单的调整来解决这个问题：用绝对访问时间戳 t 替代地点索引 o，并将参数 $\{\omega_1, \omega_2, \ldots, \omega_d\}$ 设置为可学习的。形式化表示为：

$$\boldsymbol{\Psi}(t) = [\cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_d t), \sin(\omega_d t)], \tag{4-8}$$

其中，Ψ 是 CTLE 的时间编码模块。此模块之所以能建模地点被访问时间的差异中包含的信息，是因为 $\Psi(t)$ 和 $\Psi(t+\delta)$ 之间的点积计算为：

$$\Psi(t)\cdot\Psi(t+\delta)=\cos(\omega_1\delta)+\cos(\omega_2\delta)+\cdots+\cos(\omega_d\delta), \tag{4-9}$$

这意味着 $\Psi(t)$ 和 $\Psi(t+\delta)$ 之间的距离（通过点积衡量）是可学习的，并且仅与相对差 δ 有关，而与 t 无关。在 CTLE 所使用的 Transformer 编码器中，注意力计算主要由点积操作构成，因此模型注定会捕获这种距离信息。

CTLE 的编码层将时间编码与地点嵌入相结合。具体来说，公式 (4-2) 被拓展为：

$$\boldsymbol{z}_l'=\Omega(l)+\Psi(t), \tag{4-10}$$

由此，CTLE 能够将相对访问时间差异中的时间信息融入到自监督学习和嵌入生成过程中，从而得到更高质量的嵌入向量。CTLE 编码层与时间编码相结合的整体架构如图 4-2(a) 所示。

自监督训练中的掩码小时目标：为了进一步从绝对访问时间中提取信息，CTLE 提出了掩码小时（Masked Hour, MH）自监督目标。对于轨迹 $\widetilde{\mathcal{T}}$ 中被掩盖的访问时间 t_m，MH 目标是预测其原始访问时间的小时索引：

$$\widehat{\mathrm{Hour}(t_m)}=\mathrm{FFN}_{\mathrm{MH}}(f(\widetilde{\mathcal{T}})_m), \tag{4-11}$$

其中 $\mathrm{Hour}(t)$ 表示时间 t 的小时索引。$\mathrm{FFN}_{\mathrm{MH}}$ 为全连接网络，用于将 Transformer 编码器的输出映射为对小时索引的预测。在 CTLE 的自监督训练过程中，MH 与 MLM 相结合形成多任务学习目标。MH 目标和 CTLE 整体的自监督目标形式化为：

$$\begin{aligned} O_{\mathrm{MH}}&=\arg\max_{\theta}\textstyle\sum_{t_m\in\Gamma}P(\mathrm{Hour}(t_m)|\mathrm{FFN}_{\mathrm{MH}}(f(\widetilde{\mathcal{T}})_m))\\ O&=O_{\mathrm{MLM}}+O_{\mathrm{MH}}, \end{aligned} \tag{4-12}$$

其中 θ 表示 f 的可学习参数。上述自监督目标可以抽象为 $\arg\max_\theta P(l_m,t_m|C(l_m))$，即给定上下文环境时，最大化某个目标地点在特定时间被访问的概率。如此，CTLE 将绝对访问时间信息融入到模型中。CTLE 在自监督训练阶段的整体模型架构如图 4-2(b) 所示。

4.3 实验

为了评估 CTLE 生成的上下文嵌入的质量，本节将 CTLE 应用于三个下游预测模型，并将预测准确率与其他地点嵌入方法进行比较。

4.3.1 基线模型

为了证明 CTLE 模型的优越性，实验选择了一系列地点嵌入模型作为基线模型进行比较。这些模型包括两种经典的分布式嵌入模型，以及一些最新的自监督地点嵌入模型。

- **FC Layer**：一种广泛使用的表示学习方案，为每个地点分配一个随机初始化的嵌入向量。这些向量与整个模型的其他参数一起进行训练。
- **Skip-gram**[38]：word2vec[98] 的一种实现方案，在现有工作[106] 中已被用于建模移动轨迹。
- **POI2Vec**[43]：基于 word2vec 的地点嵌入方法，通过将地点分布到地理二叉树中来建模轨迹中的空间关联性。
- **Geo-Teaser**[42]：基于 Skip-gram 的地点嵌入方法，通过扩展 Skip-gram 的输出向量和负采样策略，融合时间和空间信息。
- **TALE**[107]：本文在第 3 章中介绍的地点嵌入方法。通过设计一种新颖的时间树结构，将时间信息融入层次 Softmax 计算过程。
- **HIER**[44]：基于 LSTM[72] 和 N-gram 架构[41] 构建的动态地点嵌入模型。

值得说明的是，CTLE 应用于下游任务时会使用特殊的注意力掩码，以防止需要预测的信息泄漏给历史序列，如图 4-2(c) 所示。在地点预测训练过程中，除了 FC Layer 外的所有嵌入方法都不参与反向传播。

4.3.2 实验设置

CTLE 和各基线模型在第 2.4.3 节中定义的轨迹预测任务上进行验证。具体而言，选择适配方案（1），根据历史轨迹预测未来一个点的 POI 地点下标，序列预测模型选用 **ST-RNN**[108]、**ERPP**[109] 和 **ST-LSTM**[1]。

考虑到 CTLE 对地点功能特征的建模要求轨迹中的地点均有明确的访问语意信息，同时其对动态上下文的建模在相对完整的移动行为记录上的效果更佳，实验选用第 2.2.2 节中介绍的 Mobile-PEK 和 Mobile-SHE 手机信令数据集。对于 Mobile-PEK 数据集，前 3 天的轨迹为训练集，第 4 天为评估集，最后一天为测试集。对于 Mobile-SHE 数据集，前 7 天的轨迹为训练集，最后 2 天为测试集，其余天数为评估集。在地点预测任务中，每条轨迹的最后三个点为预测目标，其余点为源序列。

所有的地点嵌入模型和下游预测模型都使用训练集进行训练，并在评估集上验证以获得最佳的超参数。需要注意的是，自监督嵌入模型本身没有可靠的性能

指标，因此本实验中借助 ST-LSTM 模型的预测结果来调整超参数。地点预测模型使用交叉熵损失进行训练，并使用 Accuracy、macro-recall 和 macro-F1 指标评估。

PyTorch 框架[102] 被用于实现 CTLE 模型和所有下游预测模型。所有嵌入模型的嵌入向量维度 d 设置为 128。ERPP 和 ST-LSTM 基于两层 LSTM 网络实现，其中隐藏层维度设置为 512；ST-RNN 基于单层 RNN 实现，其中隐藏层维度设置为 512。CTLE 模型由 4 层 Transformer 编码器层堆叠而成，其中每层具有 8 个注意力头，全连接前馈层的隐藏维度设置为 512。CTLE 在训练集上进行总 200 轮的自监督训练。所有下游预测模型使用早停机制在评估集上找到最佳表现的轮次。模型训练过程由初始学习率为 0.0001 的 Adam 优化器控制。

4.3.3 总体性能对比

表 4-1 展示了不同嵌入模型和下游预测模型在轨迹预测任务中的性能比较结果。除了在使用 ST-RNN 预测模型的 Mobile-SHE 数据集上，CTLE 的表现均强于其他地点嵌入方法。

随机初始化的 FC Layer 仅适用于特定的预测目标，无法融入一般性的信息，例如地点的功能特征。Skip-gram 仅利用目标地点及其上下文的共现概率，忽略了时空轨迹中的独特属性，例如地点的访问时间。从表 4-1 中可以看出，这些方法通常比更全面的地点嵌入方法的预测性能更差。Geo-Teaser 和 POI2Vec 通过不同的方法将空间信息引入嵌入向量中。然而，正如第 3.3.3 节中说明的，城市中地点之间的地理相关性并不全面和准确。Geo-Teaser 还将时间信息融入了地点嵌入中。然而，它仅区分了工作日和周末发生的地点访问记录，忽略了更细粒度的时间信息。TALE 和 HIER 均考虑了地点访问时间的小时索引，可以生成更有利于预测任务的地点嵌入。

然而，上述嵌入方法都未考虑相对访问时间差异，而此类差异信息能够揭示地点间的关联性。更重要的是，它们仅分配一条潜在向量来表示每个地点，这使得它们无法准确反映地点的多功能属性。这将会严重影响下游模型的性能，因为现实世界中地点的多功能属性非常普遍且明显。

CTLE 模型根据特定的上下文环境，动态生成地点的嵌入，因此能够区分地点在特定上下文环境中的确切功能。此外，CTLE 同时考虑了轨迹的绝对访问时间信息和相对访问时间差信息。正如表 4-1 中所示，这些设计带来了更高质量的地点嵌入，并且有助于下游地点预测模型实现更好的性能。

表 4-1　不同方法间轨迹预测性能的比较

Table 4-1 Trajectory prediction performance comparison of different approaches.

预测模型		ST-RNN			ERPP			ST-LSTM		
指标		Accuracy (%)	macro-Recall (%)	macro-F1 (%)	Accuracy (%)	macro-Recall (%)	macro-F1 (%)	Accuracy (%)	macro-Recall (%)	macro-F1 (%)
数据集	嵌入模型									
Mobile-PEK	FC Layer *	3.744±0.10	1.739±0.06	1.449±0.19	4.373±0.14	2.017±0.04	1.595±0.05	4.542±0.15	2.092±0.09	1.689±0.07
	Skip-gram	3.671±0.11	1.777±0.11	1.423±0.05	4.611±0.01	2.368±0.07	1.779±0.04	4.877±0.05	2.586±0.06	1.947±0.04
	POI2Vec	3.992±0.08	2.281±0.08	1.838±0.06	5.024±0.08	2.595±0.07	2.035±0.06	5.163±0.10	2.682±0.08	2.077±0.10
	Geo-Teaser	3.998±0.13	2.166±0.07	1.796±0.07	5.159±0.05	2.671±0.08	2.039±0.03	5.305±0.05	2.739±0.04	2.084±0.02
	TALE	4.199±0.05	2.240±0.07	1.815±0.06	5.457±0.03	3.237±0.07	2.587±0.03	5.511±0.05	3.152±0.10	2.486±0.13
	HIER	4.339±0.04	2.440±0.07	1.862±0.08	5.607±0.09	2.870±0.08	2.176±0.04	5.589±0.15	2.839±0.11	2.165±0.02
	CTLE (ours)	**5.068±0.05**	**2.890±0.11**	**2.312±0.02**	**6.481±0.05**	**4.002±0.04**	**3.066±0.06**	**6.473±0.09**	**4.072±0.13**	**3.097±0.13**
Mobile-SHE	FC Layer *	3.674±0.07	2.408±0.07	1.946±0.05	4.343±0.18	2.454±0.10	2.037±0.09	4.416±0.20	2.450±0.14	2.005±0.11
	Skip-gram	3.646±0.05	2.278±0.08	1.809±0.05	4.405±0.06	2.459±0.06	1.974±0.05	4.508±0.05	2.507±0.07	1.998±0.07
	POI2Vec	3.936±0.04	2.605±0.04	2.084±0.03	4.923±0.06	2.992±0.02	2.408±0.02	4.930±0.07	2.890±0.10	2.305±0.08
	Geo-Teaser	4.006±0.05	2.455±0.03	1.897±0.02	4.932±0.12	2.895±0.02	2.410±0.07	5.130±0.15	2.754±0.07	2.245±0.06
	TALE	4.689±0.10	**3.444±0.09**	**2.761±0.08**	5.179±0.09	3.446±0.06	2.883±0.04	5.204±0.06	3.399±0.11	2.787±0.11
	HIER	4.539±0.22	3.117±0.15	2.521±0.09	5.624±0.16	3.273±0.17	2.708±0.18	5.672±0.09	3.252±0.07	2.680±0.05
	CTLE (ours)	**5.124±0.20**	3.392±0.11	2.720±0.07	**6.311±0.04**	**3.984±0.05**	**3.340±0.07**	**6.325±0.08**	**3.950±0.11**	**3.291±0.06**

* 全连接层是下游预测模型中最初使用的地点嵌入方法，因此相应行中的结果是这些模型的原始预测性能。**粗体**表示最佳结果，下划线 表示次好结果。

4.3.4 模型组件分析

为了进一步研究 CTLE 模型中各个组件的有效性，本节设计了 CTLE 模型的三种变体：

- **Basic**：仅使用来自 Transformer[73] 的位置编码模组，并仅利用 MLM 目标进行自监督训练。
- **+TempEnc**：使用 CTLE 提出的时间编码替换了 Basic 模型中的位置编码。
- **+MH**：在 Basic 模型基础上，在自监督训练过程中将掩码小时目标与 MLM 目标相结合。

这三种变体被用于与完整的 CTLE 模型在 ST-LSTM 下游预测模型上进行准确率的比较。如图 4-3 所示，即使是 Basic 模型也优于基于分布式词嵌入的地点嵌入方法。这表明学习上下文感知的地点嵌入向量对地点预测任务是有益的。时间编码模块和掩码小时目标将轨迹中的时间信息纳入模型，可以改善模型的嵌入质量。这两个模块关注时间信息的不同方面，而完整的 CTLE 模型将它们结合在一起，并获得了最佳结果。

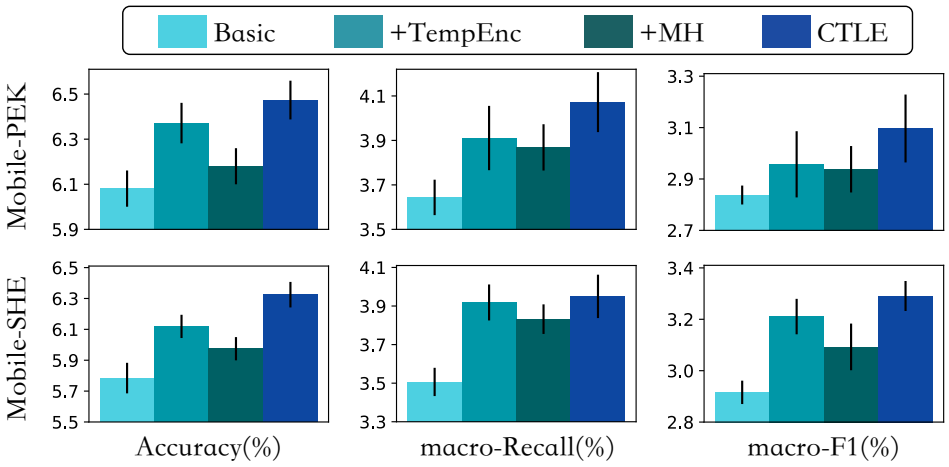


图 4-3 CTLE 模型的组件分析

Figure 4-3 Component analysis of the CTLE model.

4.4 本章小结

本章为了解决自监督学习难以考虑时空轨迹中动态上下文关联性的挑战，进行了上下文感知的轨迹自监督学习相关研究，并提出了一种新颖的上下文与时间感知的地点嵌入模型 CTLE。该模型能够在自监督学习和下游任务适配过程中，将

轨迹动态上下文关联性纳入考虑，进而得到能输出更准确的地点嵌入向量的自监督模型。

具体而言，CTLE 提出的地点嵌入映射模型基于双向 Transformer 编码器，在将目标地点映射为嵌入向量时动态地建模地点在轨迹中的上下文，使得其在适配于下游任务时能够考虑轨迹动态上下文与目标地点间的关联性。为了在自监督训练阶段使模型理解轨迹中的上下文信息，CTLE 基于掩码语言模型构建自监督目标函数。同时，CTLE 通过一个精巧的时间编码模块和一种新颖的掩码小时自监督目标，能够将多方面的轨迹访问时间信息融入到模型中。

CTLE 在两个手机信令数据集上进行验证，其自监督学习得到的模型被适配于轨迹预测任务。与基线模型的对比表明其能提升轨迹预测任务上多种下游预测模型的准确性，验证了 CTLE 自监督学习的有效性。同时，对模型的组件分析证明了 CTLE 所提出的各模块的有效性。

5　行程信息建模的轨迹自监督学习

本章旨在构建挖掘行程信息的时空轨迹自监督学习方法，提出一种起终点先验的轨迹行程生成模型 IGOP。首先，本章设计一种对轨迹噪音鲁棒、有利于模型区分轨迹异常值的轨迹行程表示形式。随后，本章提出一种基于扩散生成的自监督学习方法 IGOP。该方法能够建模轨迹行程与起终点的关联性，并能生成起终点对应的行程。IGOP 模型的自监督学习有效性和可解释性在两个轨迹数据集和两种下游任务上验证。

5.1　本章引言

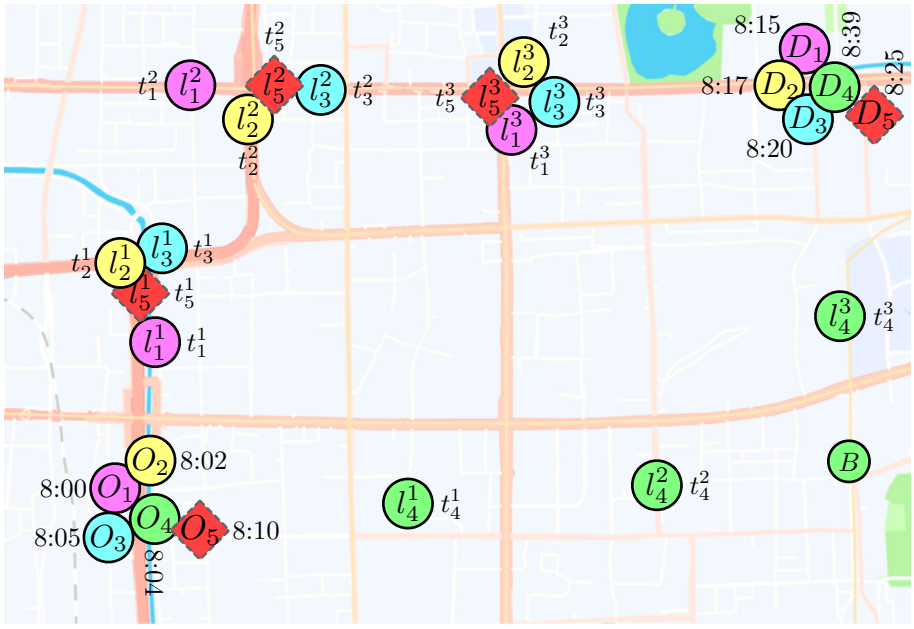


图 5-1　轨迹对应的行程与其旅行时间特征的关联性

Figure 5-1 Relationship between trajectories' itineraries and their travel times.

如第 1.2.1 节所述，时空轨迹中的行程信息能够体现其轨迹的多种时空特征，包括路线偏好和旅行时间等。这些特征能够服务于许多下游任务，包括起终点距离预测模型[110]、起终点路径预测模型[111]、外包运输服务中的定价[82,83]、整体出行成本的估算[84]、运输调度[85,86]、送货服务[87,88] 和交通流预测[89] 等。在这些任务的预测阶段，行程尚未发生，因此已知信息通常仅有第 2.1.4 节介绍的起终点，以及预计的出发时间。此已知信息可形式化定义如下。

定义 5.1 (ODT 输入，ODT-Input)**.** *ODT 输入作为未来旅程的已知信息，表示为元组 $odt = (l_o, l_d, t_o)$，聚合了起终点和出发时间。l_o 和 l_d 在此处分别是起点和终点的 GPS 坐标，t_o 表示出发时间。*

若能构建自监督学习方法，从历史的轨迹中提取行程信息并建立其与 ODT 输入的关联性，就能在未来行程未知时依然为下游任务提供丰富的特征。以图 5-1 为例，当给定未来行程的起终点 (O_5, D_5) 和出发时间 8:10 时，若自监督学习方法能够根据学习到的行程关联性推断出其行程为未来轨迹 $\mathcal{T}_5 = \langle (O_5, 8{:}10), (l_5^1, t_5^1), (l_5^2, t_5^2), (l_5^3, t_5^3), (D_5, 8{:}25) \rangle$ 对应的行程，那么该起终点的旅行时间可以被准确地估计，司机也得到了关于未来行驶路线的指导。

然而，构建自监督学习以建模轨迹行程信息具有挑战性，这主要是因为历史轨迹数据中通常包含噪音和异常值，影响模型对行程信息的建模。以图 5-1 为例，起终点 (O_1, D_1)、(O_2, D_2) 和 (O_3, D_3) 对应的轨迹 $\mathcal{T}_1$、$\mathcal{T}_2$ 和 $\mathcal{T}_3$ 拥有相同的行程，旅行时间也非常相近。自监督模型应当建模到这三条轨迹对应的行程信息，并建立行程与三对起终点的关联性。然而，从图中可以看出，由于轨迹数据包含一定噪音，各条轨迹的轨迹点均存在一定偏差，这使得将三条轨迹对应为同一行程较为困难。同时，起终点 (O_4, D_4) 对应的轨迹由于需要经过地点 B，拥有不同的行程，可以看作是一个异常值。自监督模型在建模行程关联性时应当忽略此异常值。然而，在自监督学习中使模型能够不受异常值影响具有挑战性。

为了强化自监督学习模型对历史轨迹噪音和异常值的鲁棒性，本章首先提出了一种新的行程表示形式，称为像素化轨迹（Pixelated Trajectory，PiT）。PiT 被表示为图片格式，记作 $\mathbb{R}^{N \times M \times C}$，其中 N 和 M 分别是 PiT 的高度和宽度，C 表示 PiT 中的特征数量，包括掩码（Mask）、一天中的时间（ToD）和时间偏移（Time offset）三条特征通道。PiT 的形式化定义如下。

定义 5.2 (像素化轨迹, Pixelated Trajectory)**.** *在地图上给定一个兴趣区域（Area of Interest, AOI），通常是覆盖所有历史轨迹的区域，将经度和纬度均等地分成 L_G 段，从而得到总共 ${L_G}^2$ 个空间单元。一个像素化轨迹（PiT）被表示为一个张量 $X \in \mathbb{R}^{L_G \times L_G \times C}$，其中 C 是通道的数量。每条轨迹都对应一个 PiT。*

在一个 PiT 中，$X[x, y, k]$ 记录了单元 (x, y) 中第 k 个特征的值。PiT 中包含三条特征通道，即掩码（Mask）、一天中的时间（Time of the day，ToD）和时间偏移（Time offset）。如果轨迹 $\mathcal{T}$ 中的 GPS 点从未落入单元 (x, y)，则相应单元的所有三个通道都设置为 -1。如果轨迹 $\mathcal{T}$ 中的 GPS 点 (l_i, t_i) 是第一个落入单元 (x, y) 的点，三个通道的值以如下方式计算。

1）掩码：表示轨迹是否包含落在此单元格中的 GPS 点。$X[x,y,1]=1$ 表示轨迹中至少有一个 GPS 点位于单元格 (x,y) 中。

2）一天中的时间：归一化至范围 $[-1,1]$ 内的值，表示单元格在一天内的何时被访问。计算为 $X[x,y,2]=2\times(t_i\%86400)/86400-1$，其中 t_i 表示轨迹 $\mathcal{T}$ 中第 i 个 GPS 点的 Unix 时间戳，86400 是 24 小时内的总秒数。

3）时间偏移：也是一个范围在 $[-1,1]$ 之间的归一化值，表示轨迹中此单元格时间上的访问顺序。计算为 $X[x,y,3]=2\times(t_i-t_1)/(t_{|\mathcal{T}|}-t_1)-1$，其中 t_i 表示轨迹 $\mathcal{T}$ 中第 i 个 GPS 点的 Unix 时间戳。

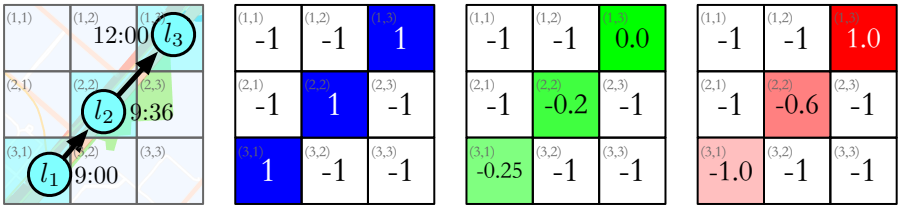


图 5-2　从轨迹构建的 PiT 的三个通道

Figure 5-2 Three channels of PiT constructed from a trajectory.

图 5-3 展示了 $\mathcal{T}_1$、$\mathcal{T}_2$、$\mathcal{T}_3$ 和 $\mathcal{T}_4$ 的 PiT 示例。示例中仅展示了 PiT 的第一个特征维度，即格式为 $X\in\mathbb{R}^{3\times4\times1}$，其中 $X[\cdot,\cdot,1]$ 为掩码，用于表示轨迹是否经过单元格。尽管 $\mathcal{T}_1$、$\mathcal{T}_2$ 和 $\mathcal{T}_3$ 存在差异，它们对应的 PiT 非常相似。这种相似性有助于模型建模多条轨迹对应的共同行程。另一方面，PiT_4 与其他 PiT 有很大的差异，让它更容易被识别为异常值，进而被移除。

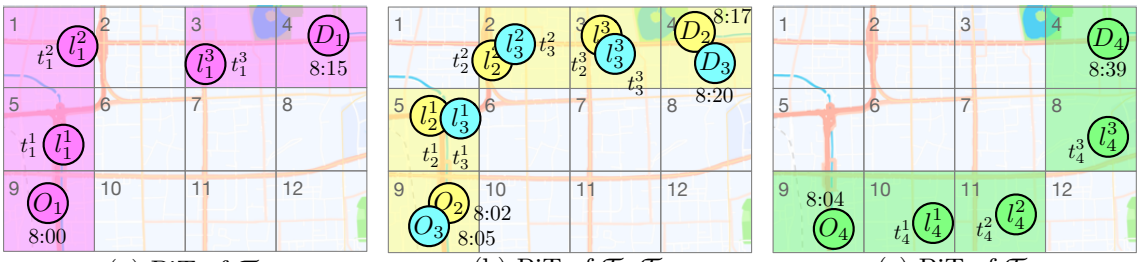


图 5-3　三条轨迹对应的 PiT 的示例

Figure 5-3 Examples of PiTs corresponding to three trajectories.

为了进一步消除异常值的影响，构建能够准确建模轨迹行程关联性的自监督学习方法，本章提出了一种新颖的**起终点先验的轨迹行程生成**（*Itinerary Generation with Origin-destination Prior*, **IGOP**）模型。IGOP 包含一个 PiT 推断模型来生成起

终点和预计出发时间对应的 PiT，并通过自监督学习训练，实现对轨迹行程和起终点关联性的建模。以图 5-1 中为例，给定起终点 (O_5, D_5) 和出发时间 8:10，IGOP 将生成轨迹 $\mathcal{T}_5$ 对应的 PiT，从而为下游任务提供准确的行程信息。为了实现 IGOP，本章提出了一个条件扩散模型，以起终点和出发时间为先验条件生成 PiT。然后，利用历史轨迹数据对该模型进行训练，使其能够建立 PiT 和起终点间的关联性。

从技术角度，IGOP 通过时空轨迹的自监督学习，得到了一个输入对象为第 2.1.4 节介绍的起终点的生成模型。该模型以起终点、出发时间为输入，生成对应的 PiT 形式的轨迹行程。

本研究内容的主要工作总结如下：

1）面向行程信息建模的轨迹自监督学习，提出了起终点先验的轨迹行程生成模型 IGOP。IGOP 能够建模轨迹行程与起终点的关联性，并对历史轨迹的噪音和异常值鲁棒。

2）提出了一种轨迹行程的表示形式 PiT，对轨迹数据中的噪音不敏感，且能帮助模型更好地区分异常值。

3）提出了一个基于扩散生成模型的 PiT 推断模型，该模型在给定起终点和出发时间的条件下，生成对应的 PiT，从而建模轨迹行程与起终点的关联性。

4）在两个轨迹数据集和两种下游任务上进行了大量实验，证明了所提出的方法在性能上优于现有方法。

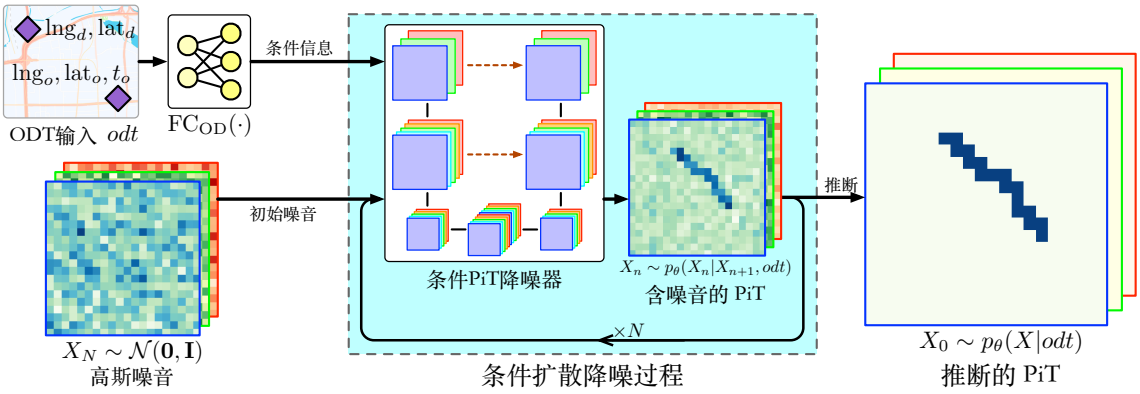


图 5-4　IGOP 模型的总体框架

Figure 5-4 The overall framework of the IGOP model.

5.2 起终点先验的轨迹行程生成模型

5.2.1 模型整体结构

本章提出了一种起终点先验的轨迹行程生成（IGOP）方法，自监督学习轨迹行程与起终点的关联性，并能生成 ODT 输入对应的轨迹行程。其总体框架如图 5-4 所示。IGOP 基于扩散生成模型，推断与给定的 ODT 输入相对应的 PiT。ODT 输入被视为条件信息，融入到一个条件 PiT 去噪器中。去噪器首先从标准高斯噪声中进行采样，然后通过多步条件去噪扩散过程生成与 ODT 输入对应的 PiT。

5.2.2 基于扩散的 PiT 推断

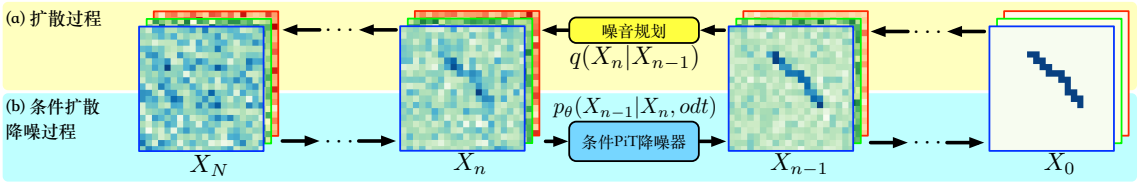


图 5-5 基于扩散的条件 PiT 推断框架中的两个马尔可夫过程

Figure 5-5 Two Markov processes in the diffusion-based conditioned PiT inference framework.

给定未来一次行程的起终点，该行程的旅行时间与驾驶员选择的路径密切相关。然而，实际的行程在旅行时间估计阶段通常是未知的。为了为下游任务提供准确的行程信息，旨在基于自监督训练，从历史行程中全面学习 ODT 输入与轨迹行程之间的关联。

假设有一条历史轨迹 $\mathcal{T}$ 及其对应的 ODT 输入，记作 $odt = (l_o, l_d, t_o)$。本节的目标是从历史轨迹集合 $\mathbb{T}$ 中学习一个后验概率 $p(\mathcal{T}|odt)$，该概率表示 odt 和 $\mathcal{T}$ 之间的关系。由于本章中使用 PiT X 表示轨迹 $\mathcal{T}$，上述后验概率可以表示为 $p(X|odt)$。然而，该概率在现实中是未知的，因此需要通过一组可学习的参数 θ 来拟合它，即将概率拟合为 $p_\theta(X|odt)$。

毫无疑问，学习模型 p_θ 对于建模准确的轨迹行程关联性至关重要。如果从 p_θ 推断出的轨迹行程与给定未来 ODT 输入的预期轨迹非常不同，那么模型提供的行程信息将不准确。本节基于去噪扩散概率模型（Denoising Diffusion Probabilistic Models, DDPM）[47] 构建该学习模型。由于 DDPM 建模的是无条件数据分布 $p(X)$，因此其生成的数据并不受先验条件限制，可以是符合训练数据集分布的任何信号。然而，本节的目标是在给定 ODT 输入的条件下对数据分布进行建模。因此，本节

提出了一种基于扩散的条件 PiT 推断框架，包括扩散过程和条件去噪扩散过程。

添加噪声到 PiT 的扩散过程： 直观地说，扩散过程会逐步向信号添加噪声，直到达到一个简单的先验分布，例如高斯分布。图 5-5(a) 清楚地展示了本节所实现的扩散过程的工作原理，即通过总共 N 个扩散步骤，从清晰的 PiT X_0 获得一个带噪声的 PiT X_N。该扩散过程的单个步骤计算为：

$$q(X_n|X_{n-1}) = \mathcal{N}(X_n; \sqrt{1-\beta_n}X_{n-1}, \beta_n\mathbf{I}), \tag{5-1}$$

其中 $\mathcal{N}(\mu;\Sigma)$ 表示具有均值 μ 和协方差矩阵 Σ 的高斯分布，β_n 是在第 n 步中用于控制噪声水平的系数。在实践中，β_n 通常对于每一步都是固定的，并由与 n 相关的单调函数控制，噪声水平随着 n 的增加而增加。本节中遵循 DDPM[47] 中使用的线性调度，即 β_n 随着 n 从 0.0001 线性缩放到 0.02。

从清晰的 PiT X_0 开始，可以通过以下方式获得第 n 步中带噪声的 PiT X_n：

$$q(X_n|X_0) = \prod_{m=1}^{n} q(X_m|X_{m-1}) \tag{5-2}$$

由于每个步骤中的噪声水平是固定的，并且添加的噪声遵循高斯分布，可以利用高斯分布的性质将公式 (5-2) 简化为以下形式。

$$q(X_n|X_0) = \mathcal{N}(X_n; \sqrt{\overline{\alpha}_n}X_0, (1-\overline{\alpha}_n)\mathbb{I}), \tag{5-3}$$

其中 $\alpha_n = 1-\beta_n$，$\overline{\alpha}_n = \prod_{m=1}^{n}\alpha_m$。因此，在扩散过程中，可以通过将特定的噪声添加到原始的 X_0 直接得到 X_n，而无需逐步添加噪声。

扩散过程的最后一步的输出应当符合标准高斯分布，表示如下。

$$X_N \sim q(X_N) = \mathcal{N}(X_N; \mathbf{0}, \mathbf{I}) \tag{5-4}$$

X_N 是后续 PiT 推断过程的一个良好起点，因为它不包含任何先验信息。换句话说，模型可以从标准高斯分布中随机采样噪声，作为推断过程的起始点。

基于条件去噪扩散过程的 PiT 推断： 为了根据未来的 ODT 输入推断出对应的 PiT，本节实现 ODT 输入先验下的逆扩散过程。更具体地说，该过程逐渐从带噪声的 PiT 中移除噪声，直到获得与 ODT 输入对应的无噪声 PiT。这个过程被称为条件去噪扩散过程，图 5-5(b) 描绘了这一过程。条件去噪扩散过程的单个步骤可以表示如下。

$$p(X_{n-1}|X_n, odt) = \mathcal{N}(X_{n-1}; \boldsymbol{\mu}_{n-1}, \boldsymbol{\Sigma}_{n-1}), \tag{5-5}$$

其中 $\boldsymbol{\mu}_{n-1}$ 和 $\boldsymbol{\Sigma}_{n-1}$ 是第 $n-1$ 步的均值和方差，X_{n-1} 遵循条件概率，该概率依赖于前一步的带噪声 PiT X_n 和 ODT 输入 odt。然而，在实际中，p 的形式是未知的，一

般需要两个神经网络模块来拟合均值和方差。因此，公式 (5-5) 可以重新表达如下。

$$p_\theta(X_{n-1}|X_n, odt) = \mathcal{N}(X_{n-1}; \boldsymbol{\mu}_\theta(X_n, n, odt), \boldsymbol{\Sigma}_\theta(X_n, n, odt)), \tag{5-6}$$

其中 $\boldsymbol{\mu}_\theta(\cdot)$ 和 $\boldsymbol{\Sigma}_\theta(\cdot)$ 表示由参数 θ 控制的神经网络估计的均值和方差。完整的 PiT 推断过程可以表示如下。

$$p_\theta(X_0|X_N, odt) = \prod_{n=1}^{N} p_\theta(X_{n-1}|X_n, odt) \tag{5-7}$$

如公式 (5-4) 所示，X_N 可以从标准高斯分布中采样得到，而 odt 则来自于 ODT-Oracle 问题的输入。算法 5.1 详细说明了基于这两个输入推断 PiT 的过程。

由于概率模型被用于推断给定 ODT 输入最可能对应的 PiT，因此异常值的影响可以被弱化。

算法 5.1 : 给定 ODT 输入推断 PiT

输入: 概率 $p_\theta(X_{n-1}|X_n, odt)$, 未来 ODT 输入 odt

输出: odt 对应的 PiT X

初始化 $n \leftarrow N, X_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;

while $n > 0$ **do**

  采样 $X_{n-1} \sim p_\theta(X_{n-1}|X_n, odt)$;

  设置 $X_n \leftarrow X_{n-1}, n \leftarrow n-1$;

end

获取生成结果 X_0 作为 odt 对应的 PiT X;

PiT 推断模型的重参数化与训练： 为了简化参数 θ 的训练过程，本节遵循 DDPM[47] 的做法，固定公式 (5-6) 中的方差来构建条件去噪扩散过程，即 $\Sigma_\theta(X_n, n, odt) = \sqrt{\beta_n}\mathbf{I}$。因此，模型只需估计均值。

同时，本节对均值 $\boldsymbol{\mu}_\theta(\cdot)$ 进行重参数化。与直接预测 $\boldsymbol{\mu}_\theta(\cdot)$ 不同，重参数化后的模型旨在预测扩散过程中第 n 步添加的噪声。将预测的噪声记为 $\varepsilon_\theta(X_n, n, odt)$，则 $\boldsymbol{\mu}_\theta(\cdot)$ 可以重参数化如下。

$$\boldsymbol{\mu}_\theta(X_n, n, odt) = \frac{1}{\sqrt{\alpha_n}}(X_n - \frac{\beta_n}{\sqrt{1-\overline{\alpha}_n}}\varepsilon_\theta(X_n, n, odt)) \tag{5-8}$$

接下来，重参数化公式 (5-6)，并将 $\boldsymbol{\mu}_\theta(\cdot)$ 替换为公式 (5-8)，可以得到以下去噪过程的形式化。

$$\begin{aligned} X_{n-1} &= \boldsymbol{\mu}_\theta(X_n, n, odt) + \boldsymbol{\Sigma}_\theta(X_n, n, odt)\varepsilon \\ &= \frac{1}{\sqrt{\alpha_n}}(X_n - \frac{\beta_n}{\sqrt{1-\overline{\alpha}_n}}\varepsilon_\theta(X_n, n, odt)) + \sqrt{\beta_n}\varepsilon, \end{aligned} \tag{5-9}$$

其中 $\varepsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})$。

为了训练 PiT 推断模型，需要对条件去噪扩散过程的所有步骤进行监督。在实际的训练过程中，可以从均匀分布 $U(1,N)$ 中采样 n，以确保最终能够训练所有步骤。在第 n 步，使用均方误差来最小化真实噪声和预测噪声之间的差异，损失函数可以如下形式化。

$$\begin{aligned} L_n &= \| \varepsilon - \varepsilon_\theta(X_n, n, odt) \|^2 \\ &= \| \varepsilon - \varepsilon_\theta(\sqrt{\overline{\alpha}_n} X_0 + \sqrt{1-\overline{\alpha}_n}\varepsilon, n, odt) \|^2, \end{aligned} \tag{5-10}$$

其中，噪声 PiT X_n 由公式 (5-3) 中呈现的简化形式计算。详细的训练过程见算法 5.2。

算法 5.2 : 训练 PiT 推断模型

输入: 带有一组可学习参数 θ 的噪音预测器 ε_θ

输出: 训练后的参数 θ

while 模型未收敛 **do**
- 采样 $\mathcal{T} \in \mathbb{T}$，计算 ODT 输入 odt，将对应的 PiT X 作为初始图片 X_0;
- 采样 $n \sim \text{Uniform}(1,2,\ldots,N)$;
- 采样 $\varepsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})$;
- 计算 $X_n = \sqrt{\overline{\alpha}_n} X_0 + \sqrt{1-\overline{\alpha}_n}\varepsilon$;
- 采取梯度下降步骤 $\bigtriangledown_\theta \| \varepsilon - \varepsilon_\theta(X_n, n, odt) \|^2$;

end

5.2.3 条件 PiT 去噪器

本节详细描述去噪器 $\varepsilon_\theta(X_n, n, odt)$ 的实现。去噪器需要满足两个基本要求：第一个要求是预测的噪声，即去噪器的输出，应该与输入的噪声 PiT X_n 的张量尺寸相同；第二个要求是去噪器以噪声 PiT X_n、步骤 n 和 ODT 输入 odt 作为输入。在满足这些要求的基础上，本节基于 Unet[112] 实现了去噪器，这是计算机视觉领域广泛使用的神经网络。Unet 由于其瓶颈结构和残差连接设计，在高效性方面具有优势。然而，原始的 Unet 无法接受 n 和 odt 作为输入。因此，本节将这些特征转化到潜在空间内，并将它们融合到基于 Unet 的条件 PiT 去噪器中。本节提出的去噪器的总体架构如图 5-6(a) 所示。

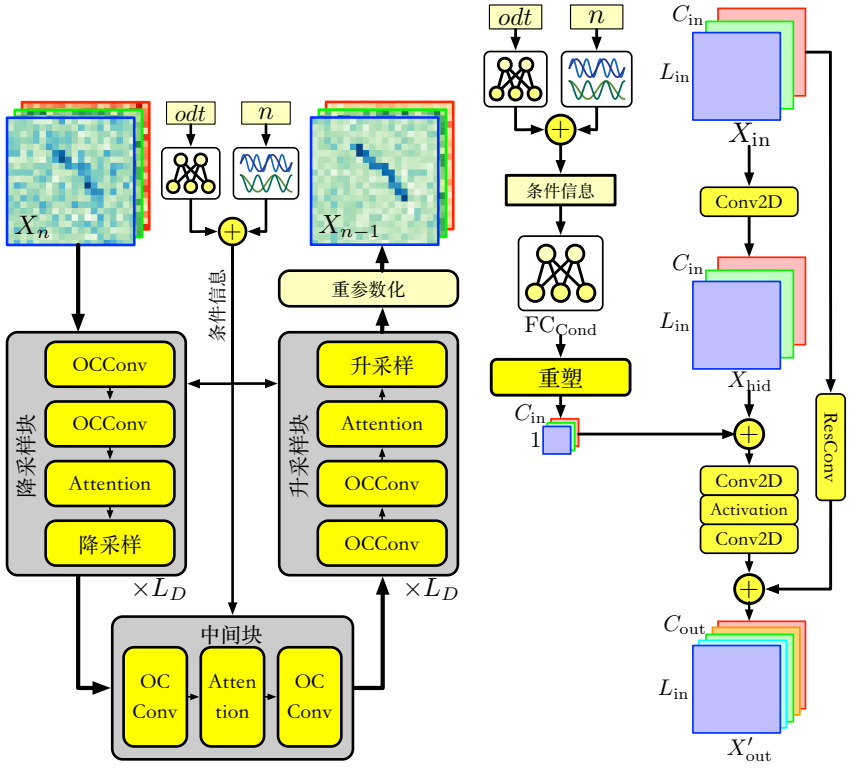
首先，步骤 n 被位置编码层映射为嵌入向量 $\text{PE}(n) \in \mathbb{R}^d$。具体的编码计算方式为：

$$\begin{aligned} \text{PE}(n)[2i] &= \sin(n/10000^{2i/d}) \\ \text{PE}(n)[2i-1] &= \cos(n/10000^{2i/d}), \end{aligned} \tag{5-11}$$

其中，d 是一个偶数值，$i \in \{1,\ldots,d/2\}$ 是特征向量的维度指示器。

同时，全连接层将 odt 转化到 d 维的潜在空间中，具体形式如下所示。

$$\mathrm{FC}_{\mathrm{OD}}(odt) : \mathbb{R}^5 \to \mathbb{R}^d \tag{5-12}$$



(a) PiT 降噪器的整体结构　　(b) OCConv 模块中条件信息的融入

图 5-6　条件 PiT 降噪器的结构与信息融入设计

Figure 5-6 The architecture of the conditioned PiT denoiser and its information fuse design.

接下来将介绍如何将这些潜在表示融合到本节所提出的 PiT 去噪模型中。遵循 Unet[112] 的架构，本节的去噪模型由 L_D 个下采样块、一个中间块和 L_D 个上采样块构成，在下采样块和上采样块之间有残差连接。每个下采样块和上采样块包含顺序堆叠的多个模块：两层基于 ConvNeXt[113] 构建的ODT-Input Conditioned Convolutional (OCConv) 模块，一层多头点积注意力模块，以及一层用于上采样或下采样的卷积模块。中间块为中间夹着一层注意力模块的两层 OCConv 模块。下采样块在空间上进行下采样，但在特征通道上进行扩展，而上采样块则相反。去噪模型的每个 OCConv 模块中都融入了条件信息 odt 和 n，数据融合流程如图 5-6(b) 所示。

具体来说，OCConv 模块以图像格式的张量作为输入，表示为 $X_{\mathrm{in}} \in \mathbb{R}^{L_{\mathrm{in}} \times L_{\mathrm{in}} \times C_{\mathrm{in}}}$。

输入由二维卷积层 Conv2D 进行处理，保持所有维度不变，得到隐藏状态 X_{hid}。

$$X_{\text{hid}} = \text{Conv2D}(X_{\text{in}}),\ X_{\text{hid}} \in \mathbb{R}^{L_{\text{in}} \times L_{\text{in}} \times C_{\text{in}}} \tag{5-13}$$

然后，全连接层将 $\text{PE}(n)$ 和 $\text{FC}_{\text{OD}}(odt)$ 的特征之和转换成维度为 C_{in} 的向量，然后该向量被添加到 X_{hid} 中的每个像素上，如下所示。

$$\begin{aligned} &\text{FC}_{\text{Cond}}(\cdot) : \mathbb{R}^{d} \to \mathbb{R}^{C_{\text{in}}} \\ &X'_{\text{hid}}[:,:,i] = X_{\text{hid}}[:,:,i] + \text{FC}_{\text{Cond}}(\text{PE}(n) + \text{FC}_{\text{OD}}(odt))[i], \end{aligned} \tag{5-14}$$

其中 $i = 1, 2, \ldots, C_{\text{in}}$。

最后，X'_{hid} 经过具有激活函数和残差连接的双层二维卷积网络，得到输出状态：

$$\begin{aligned} X_{\text{out}} &= \text{Conv2D}(\sigma(\text{Conv2D}(X'_{\text{hid}}))) \\ X'_{\text{out}} &= X_{\text{out}} + \text{ResConv}(X_{\text{in}}), \end{aligned} \tag{5-15}$$

其中 $\sigma(\cdot)$ 是激活函数，这里使用高斯误差线性单元（GELU）[114]。ResConv 是使用二维卷积实现的残差连接。输出状态 $X'_{\text{out}} \in \mathbb{R}^{L_{\text{in}} \times L_{\text{in}} \times C_{\text{out}}}$ 随后输入到降噪器中的后续模块。通常情况下，对于下采样块，$C_{\text{out}} = 2 \cdot C_{\text{in}}$；对于上采样块，$C_{\text{out}} = \lfloor C_{\text{in}}/2 \rfloor$。

5.2.4 基于 PiT 的旅行时间预测

在 IGOP 的自监督训练完成后，所得到的生成模型可直接用于起终点行程生成任务。为了扩展模型的下游任务适配性，本节介绍一种将 IGOP 高效地适配于起终点旅行时间估计的方案。图 5-7 展示了该方案的流程，IGOP 推断得到的 PiT 经过处理后送入预测模型，得到对旅行时间的估计。

由于推断得到的 PiT X 以像素化形式表示，很自然地可以想到基于卷积神经网络（Convolutional Neural Network, CNN）的估计器。然而，CNN 侧重于建模局部特性，而在旅行时间估计中，捕捉全局相关性是至关重要的。视觉 Transformer（Vision Transformer, ViT）[115] 利用自注意力来考虑像素之间的全局相关性，似乎非常适合本节的情况。然而，PiT 中的许多像素不包含有效信息，因此原始的 ViT 在这里表现不佳。为此，本节提出了遮蔽视觉 Transformer（Masked Vision Transformer, MViT）来显著提高训练和预测的效率。

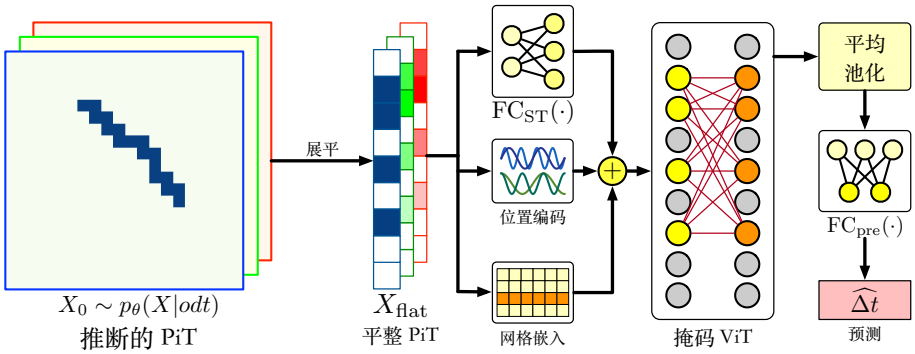


图 5-7　基于 PiT 的旅行时间预测

Figure 5-7 Travel time estimation based on PiT.

(1) PiT 扁平化和特征提取

给定推断得到的 PiT $X \in \mathbb{R}^{L_G \times L_G \times C}$，首先将其扁平化为一个长度为 $L_G{}^2$ 的序列：

$$\begin{aligned} X_{\mathrm{flat}} = \langle & X[1,1,:], X[1,2,:], \ldots, X[1,L_G,:], \\ & X[2,1,:], X[2,2,:], \ldots, X[2,L_G,:], \\ & \ldots \\ & X[L_G,1,:], X[L_G,2,:], \ldots, X[L_G,L_G,:] \rangle \end{aligned} \tag{5-16}$$

在扁平化后，PiT 中的单元 (x,y) 变成了序列中的第 $(x+(y-1)\times L_G)$ 项。此外，X_{flat} 的排列方式遵循像素的排列，而不像时间序列那样遵循时间顺序。

接着，本节进一步使用三种嵌入模块从 X_{flat} 中提取时空特征：

1）*单元嵌入模块* $E[\cdot]$。初始化一个嵌入矩阵 $E \in \mathbb{R}^{L_G{}^2 \times d_E}$，其中第 $x+(y-1)\times W$ 列是单元 (x,y) 的嵌入向量，即扁平序列中项目 $X[x,y,:]$ 的嵌入向量；d_E 是嵌入维度。E 可以看作是单元的固有空间特征。

2）*位置编码模块* $\mathrm{PE}(\cdot)$。由于自注意力是无序的，使用公式 (5-11) 中引入的位置编码来对扁平序列中的项目位置进行编码。其中，项目 $X[x,y,:]$ 的编码向量是 $\mathrm{PE}(x+(y-1)\times W) \in \mathbb{R}^{d_E}$。

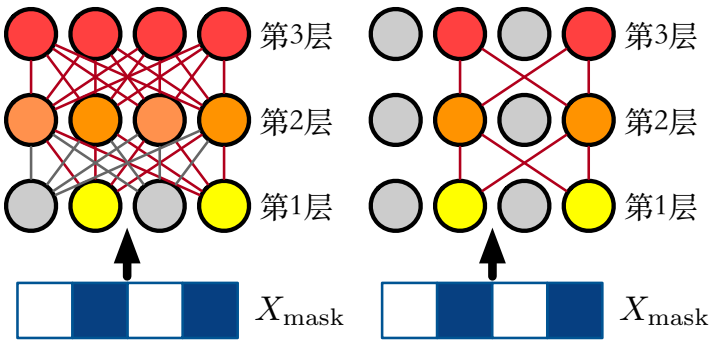
3）*潜在空间转换模块* $\mathrm{FC}_{\mathrm{ST}}(\cdot)$。PiT 的每个单元包含三个特征通道。本模块使用全连接层将它们转换到潜在空间中：$\mathrm{FC}_{\mathrm{ST}}(X[x,y,:]) : \mathbb{R}^3 \to \mathbb{R}^{d_E}$。

三种模块的输出相加形成 X_{flat} 中每一项的潜在输入向量：

$$\begin{aligned} X_{\mathrm{latent}}[x,y] = & E[x+(y-1)\times W] + \\ & \mathrm{PE}(x+(y-1)\times W) + \\ & \mathrm{FC}_{\mathrm{ST}}(X[x,y,:]) \end{aligned} \tag{5-17}$$

接着，潜在序列 $X_{\text{latent}} = \langle X_{\text{latent}}[1,1], X_{\text{latent}}[1,2], \cdots, X_{\text{latent}}[L_G, L_G] \rangle$ 被作为 MViT 模型的输入。

(2) 掩码视觉 Transformer



(a) 原始 Vision Transformer　　(b) 掩码 Vision Transformer

图 5-8　原始 ViT 与掩码 ViT 的注意力掩码方案比较

Figure 5-8 Comparison of attention mask scheme between vanilla ViT and Masked ViT.

X_{latent} 中的许多项目不包含任何有效信息，因为它们在 PiT 中的相应特征被设置为 -1。在 ViT 中，可以应用注意力掩码使这些项目的注意力权重被设置为零。然而，这种掩码方案不能提高计算效率，因为所有项目的注意力权重仍然会被计算，如图 5-8(a) 所示。这对于本研究的情况来说问题尤为明显，因为 PiT 覆盖了整个 AOI，通常非常稀疏。

为了加速对扁平化 PiT 序列的计算，本节实现了一种更高效的掩码方案，提出了掩码视觉 Transformer（Masked Vision Transformer, MViT）。首先，通过第一个通道的 PiT 来计算掩码，以确定 X_{latent} 中的项目是否包含有效信息：

$$X_{\text{mask}}[x,y] = \begin{cases} \text{True} & \text{如果 } X[x,y,1] \geqslant 0 \\ \text{False} & \text{如果 } X[x,y,1] < 0 \end{cases}, \tag{5-18}$$

其中，$x \in \{1,\ldots,L_G\}$，$y \in \{1,\ldots,L_G\}$。因此，如果 $X_{\text{mask}}[x,y] = \text{True}$，则 $X_{\text{latent}}[x,y,:]$ 包含有效信息。

类似于 Transformer[73] 和 ViT，MViT 由多层堆叠而成。其中，每层包含两个模块：自注意力模块和前馈神经网络，两者均采用残差连接。与 ViT 不同，MViT 中的自注意力仅应用于包含有效信息的项目。这等价于仅保留具有有效信息的项目来形成一个掩码序列，并仅对掩码序列应用自注意力，如图 5-8(b) 所示。形式

上，一层 MViT 的计算如下：

$$X_{\text{out-seq}} = \text{FFN}(\text{Att}(\text{Mask}(X_{\text{in-seq}}, X_{\text{mask}}))) \tag{5-19}$$

$$\begin{aligned}\text{Mask}(X_{\text{in-seq}}, X_{\text{mask}}) = \langle & X_{\text{in-seq}}[x, y, :], \\ & x \in \{1, \ldots, L_G\}, y \in \{1, \ldots, L_G\}, \\ & X_{\text{mask}}[x, y] = \text{True} \rangle,\end{aligned} \tag{5-20}$$

其中，$X_{\text{in-seq}}$ 和 $X_{\text{out-seq}}$ 分别表示 MViT 层的输入序列和输出序列，$\text{Mask}(X_{\text{in-seq}}, X_{\text{mask}})$ 表示掩码输入序列，$\text{Att}(\cdot)$ 和 $\text{FFN}(\cdot)$ 分别表示多头注意力和前馈神经网络。由于 MViT 仅将自注意力应用于具有有效信息的项目，计算成本取决于有效项目的总数，而不是完整序列的长度。因此，MViT 可以显著提高扁平化 PiT 序列的计算效率。

然后，MViT 由 L_E 层堆叠而成，其计算过程形式化为：

$$X'_{\text{latent}} = \text{MViT}(X_{\text{latent}}, X_{\text{mask}}), \tag{5-21}$$

其中记忆序列 X'_{latent} 的维度与 X_{latent} 相同，其长度等于 X_{latent} 中有效项目的数量。

接下来，平均池化层和全连接层被用于计算旅行时间预测：

$$\widehat{\Delta t} = \text{FC}_{\text{pre}}(\text{mean}(X'_{\text{latent}})), \tag{5-22}$$

其中池化操作应用于序列长度的维度。

最后，为了训练 PiT 旅行时间估计模型，使用均方差作为损失函数，以使预测结果尽可能接近真实值，具体如下。

$$L_{\text{pre}} = ||\Delta t - \widehat{\Delta t}||^2 \tag{5-23}$$

通过合理地利用推断出的 PiT 进行旅行时间估计，结合本节提出的预测模型，IGOP 能够在不依赖未来轨迹数据的情况下实现高准确度的旅行时间估计。

5.3 实验

为了评估所提出的 IGOP 框架的有效性，本节在两个真实轨迹数据集上进行了大量实验，并将 IGOP 与现有方法进行了比较。

5.3.1 基线方法

IGOP 与十一种基线方法进行比较。其中两种是起终点行程生成方法，两种是旅行时间估计方法，其余的则是起终点旅行时间估计方法。

起终点行程生成方法： 这些方法从路网中识别从起点到终点的最佳路径。实验中使用加权路网，其中权重表示从历史轨迹计算得出的道路段的平均行驶时间。输出的旅行时间是所识别路径中道路段的历史平均行驶时间之和。

- **Dijkstra**[21]：一种最短路径算法，计算起点到终点的最小权重路径。
- **DeepST**[60]：基于对历史出行行为的学习，生成起点到终点最可能的行驶路径。

轨迹旅行时间估计方法： 这些方法根据给定的行驶路径预测旅行时间。由于在起终点旅行时间估计场景中真实的行驶路径未知，实验中使用由 DeepST 生成的路径作为这些方法的输入。

- **WDDRA**[116]：利用多任务辅助损失来提高行驶时间预测的准确性。
- **STDGCN**[6]：利用神经网络结构搜索技术自动识别最佳网络结构。

起终点旅行时间估计方法： 这些方法旨在基于 ODT 输入预测旅行时间。

- **TEMP**[117]：对具有相近出发地、目的地和出发时间的历史轨迹的旅行时间进行加权平均。
- **LR**：从历史行程中学习从 ODT 输入到旅行时间的线性映射。
- **GBM**：一种非线性回归方法，使用 XGBoost[118] 实现。
- **RNE**[119]：在嵌入空间中计算路网节点之间的最短路径距离。
- **ST-NN**[120]：联合预测给定出发地和目的地的旅行距离和时间。
- **MURAT**[121]：联合预测给定出发地、目的地和出发时间的行驶距离和时间。
- **DeepOD**[122]：借助训练阶段的辅助损失，将 ODT 输入与历史轨迹之间的相关性引入模型。

5.3.2 实验设置

IGOP 和各基线模型在 2.4.3 节中介绍的起终点行程生成和起终点旅行时间估计任务上验证。考虑到相对稠密采样的轨迹数据能更好地体现行程信息，实验选用 2.2.3 中介绍的成都和哈尔滨出租车定位数据集。所有数据集首先按照出发日期和时间对轨迹进行排序，然后按 8:1:1 的比例划分为训练集、验证集和测试集。PiT 推断模型在训练集上进行了 50 轮训练，随后在验证集和测试集上进行 PiT 推断。PiT 行驶时间估计在训练集上进行训练，在验证集上实现早停机制，并在测试集上计算最终的评估指标。起终点旅行时间估计方法的准确性使用均方根误差

（RMSE）、平均绝对误差（MAE）和平均绝对百分比误差（MAPE）来评估。

所有方法均使用 Python 和 PyTorch[102] 进行实现。基线方法按照其原始论文中建议的参数进行实现。对于提出的 IGOP 模型的超参数，本节考虑不同模块中的 5 个关键参数。它们的范围和最优值列在表 5-1 中。值得一提的是，所有超参数都是基于验证集上的 MAE 结果进行选择的。稍后还将在测试集上验证这些超参数的有效性。

表 5-1　超参数范围和最佳值

Table 5-1 Hyper-parameter range and optimal values.

参数	范围
L_G	10, 15, 20, 25, 30
N	500, 1000, 1500, 2000
L_D	1, 2, 3, 4
d_E	32, 64, 128, 256
L_E	1, 2, 3, 4

下划线 表示最佳参数设置。

模型训练使用 Adam 优化器和 0.001 的初始学习率。所有实验在配备了 Intel(R) Xeon(R) W-2155 CPU 和 nVidia(R) TITAN RTX GPU 的 Ubuntu 20.04 服务器上运行。

5.3.3　总体性能对比

(1) 起终点行程生成性能对比

表 5-2　IGOP 模型的 PiT 推断准确率

Table 5-2 PiT inference accuracy of the IGOP model.

数据集	成都 / 哈尔滨	
指标	RMSE	MAE
Overall	0.196/0.181	0.027/0.023
Channel 1 (Mask)	0.271/0.224	0.039/0.028
Channel 2 (ToD)	0.128/0.183	0.016/0.024
Channel 3 (Offset)	0.159/0.123	0.025/0.016

表 5-3　不同方法的路线推断准确率比较

Table 5-3 Comparison of route inference accuracy between difference approaches.

数据集	成都 / 哈尔滨		
指标	Pre(%)	Rec(%)	F1(%)
Dijkstra	68.918/45.459	31.310/42.525	42.065/39.993
DeepST	59.755/74.519	55.776/62.907	56.911/66.029
IGOP	**87.890/88.190**	**88.684/88.982**	**88.280/88.584**

粗体表示最佳结果，下划线 表示次好结果。

为了评估起终点行程生成的准确率，本节在给定测试集 ODT 输入的情况下，计算模型推断的 PiT 和真实值之间的 RMSE 和 MAE，结果展示在表 5-2 中。同时，本节还将 IGOP 推断的行程与 Dijkstra 和 DeepST 规划的行程的准确性进行比较。这些方法规划的路线被转化为 PiT 中第一个通道的形式，并计算 Precision (Pre), Recall (Rec) 和 f1 score (F1) 指标的值，结果展示在表 5-3 中。

可以观察到，IGOP 在两个数据集上均实现了相当准确的 PiT 推断，这有助于在第二阶段的旅行时间估计中取得较好的性能。同时，准确的路线推断意味着在给定未来旅行计划的情况下，IGOP 可以为用户提供大多数驾驶员最可能选择的路线。

(2) 起终点旅行时间估计性能对比

表 5-4 比较了不同方法的起终点旅行时间估计准确率。本章提出的 IGOP 方法在两个数据集上始终表现最佳。

在旅行时间估计方面，路径规划方法的性能在很大程度上取决于计算出的路径准确性。Dijkstra 专注于寻找最短路径，这导致了较低的旅行时间估计准确性。相比之下，DeepST 从历史轨迹数据中学习行程行为，从而显著提高了估计准确性。

WDDRA 和 STDGCN 在行程路径来源 DeepST 的基础上略微提高了性能，因为它们能够学习行程路径和行程时间之间的复杂关系。STDGCN 比 WDDRA 具有更高的准确性，这得益于神经架构搜索技术。

四种传统的起终点旅行时间估计方法，TEMP、LR、GBM 和 RNE，通过特征工程和模型设计来解决起终点旅行时间估计问题。这些方法在建模 ODT 输入和旅行时间之间的复杂关系方面面临困难，并且它们未考虑历史轨迹。其中，线性方法 LR 的表现最差，这表明 ODT 输入和旅行时间之间的相关性是非线性的。历史

平均方法 TEMP 的结果居第二，主要是由于不同 ODT 输入下历史行程的不平衡以及历史轨迹中的异常值。GBM 的表现优于 TEMP 和 LR，这归因于它相对较高的模型容量。RNE 在传统的起终点旅行时间估计方法中表现最佳，因为它使用了分层嵌入以更好地捕捉地点之间的距离。

四种基于神经网络的起终点旅行时间估计方法始终优于传统方法。这表明基于神经网络的方法在提取 ODT 输入和旅程时间之间复杂的时空关联性方面表现出色，得益于其更高的模型容量。因此，它们能够学习更合适的时空特征表示。比较输入特征相同的 GBM 与 ST-NN 模型，这个结论更加明确：可以观察到 ST-NN 的结果比 GBM 要好得多。还可以观察到 MURAT 在两个数据集上在所有指标上都优于 ST-NN，这是因为 MURAT 考虑了更全面的因素，例如路网、空间和时间。然而，上述方法均未考虑历史轨迹。可以观察到 DeepOD 在大多数情况下获得了第二好的结果，但仍然不及 IGOP。这表明不适当处理历史轨迹中的异常值可能会损害起终点旅行时间估计的性能。

表 5-4　不同方法总体的起终点旅行时间估计性能

Table 5-4 Overall origin-destination travel time estimation performance of different approaches.

数据集	成都 / 哈尔滨		
指标	RMSE (分钟)	MAE (分钟)	MAPE (%)
Dijkstra	9.677/11.865	7.618/8.447	48.618/55.261
DeepST	4.717/8.926	3.452/5.849	27.503/37.772
WDDRA	4.581/8.836	3.210/5.705	24.553/35.617
STDGCN	4.469/8.679	3.104/5.564	23.187/<u>33.771</u>
TEMP	5.578/10.150	4.267/7.891	36.611/66.781
LR	6.475/10.290	5.036/8.006	44.514/67.669
GBM	4.999/9.069	3.655/6.748	29.636/54.413
RNE	4.624/8.571	3.416/6.245	27.660/47.956
ST-NN	3.961/8.492	2.803/6.114	21.532/45.891
MURAT	<u>3.646</u>/7.937	2.384/5.360	18.345/41.128
DeepOD	3.764/<u>7.859</u>	<u>1.789</u>/<u>4.533</u>	<u>14.997</u>/36.974
IGOP	**3.177/7.462**	**1.272/3.213**	**11.343/26.698**

粗体表示最佳结果，<u>下划线</u> 表示次好结果。

本章所提出的 IGOP 方法在两个数据集上均取得了最佳结果。IGOP 将原始轨迹转换为像素化轨迹（PiT），以使模型对轨迹中局部的小差异更加鲁棒。在自监

督训练过程中，IGOP 明确地对 ODT 输入和历史行程的 PiT 之间的相关性进行建模，准确地建模轨迹行程与起终点的关联性。在旅行时间估计过程中，IGOP 根据未来的 ODT 输入推断出最可能的 PiT，并利用推断出的 PiT 进行准确的行程时间估计。IGOP 的优越性证明了考虑历史轨迹中噪音和异常值的影响的必要性。

(3) 异常值鲁棒性对比

为了进一步验证 IGOP 对噪音和异常值的鲁棒性，将 IGOP 与结合了异常值去除方法的基线模型进行对比。具体而言，实现了目前最先进的异常值检测方法 DeepTEA[9]，从轨迹数据集中去除异常值。然后，重新训练一组基线方法，并评估它们的旅行时间估计性能。结果见表 5-5。除了成都数据集上的 RNE 之外，所有基准方法都获得了性能改进。这证明，对历史轨迹中的异常值进行适当处理有助于实现更准确的轨迹行程关联性建模。然而，即使在去除异常值后，IGOP 仍然能够胜过基线方法。这是因为所提出的 PiT 表示以及所使用的扩散生成模型[47,123] 使 IGOP 能够更有效地识别并应对轨迹数据中的异常值，同时保证对噪音的鲁棒性。

表 5-5　基线模型在应用异常值去除后的性能
Table 5-5 Performance of baselines with outlier removal.

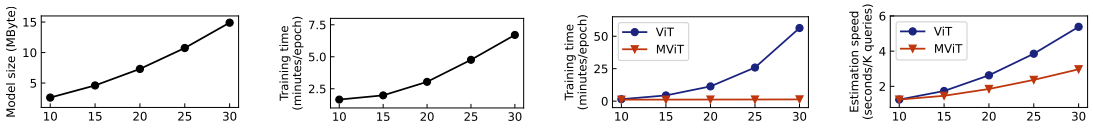
数据集	成都 / 哈尔滨		
指标	RMSE (分钟)	MAE (分钟)	MAPE (%)
Dijkstra+DeepTEA	9.641/11.862	7.582/8.396	48.337/53.949
DeepST+DeepTEA	4.692/8.901	3.416/5.821	26.959/37.063
WDDRA+DeepTEA	4.497/8.584	3.140/5.545	23.537/34.723
STDGCN+DeepTEA	4.393/8.569	3.056/5.501	22.812/33.688
RNE+DeepTEA	4.627/8.403	3.447/6.061	28.239/45.345
ST-NN+DeepTEA	3.912/8.427	2.740/5.994	20.818/43.664
MURAT+DeepTEA	3.644/7.899	2.367/5.181	17.986/37.728
DeepOD+DeepTEA	3.763/7.817	1.783/4.345	14.835/33.127
IGOP	**3.177/7.462**	**1.272/3.213**	**11.343/26.698**

粗体表示最佳结果，下划线 表示次好结果。

5.3.4 模型分析

(1) 格网尺寸的影响

为了研究 PiT 分辨率的影响，本实验改变格网尺寸 L_G，并研究模型大小、自监督训练和旅行时间估计的计算速度。从图 5-9(a) 和 5-9(b) 中，可以观察到模型大小和自监督训练的时间随着 L_G 的增加而增加。这是因为，当 L_G 增大时，PiT 变得更大，导致了条件化 PiT 去噪器中的内核和滤波器的尺寸增加。图 5-9(c) 显示，在旅行时间估计中，与原始 ViT 相比，所提出的 MViT 与 L_G 的训练用时显著降低。从图 5-9(d) 中可以看出，与 ViT 相比，MViT 在估计速度方面也有显著提升。总结来说，所提出的 MViT 方法可以提高旅行时间估计的效率。



(a) 模型大小　(b) 自监督训练速度　(c) 旅行时间估计训练速度　(d) 预测速度

图 5-9　网格长度 L_G 对效率的影响

Figure 5-9 Efficiency impact of grid length L_G.

(2) 超参数的影响

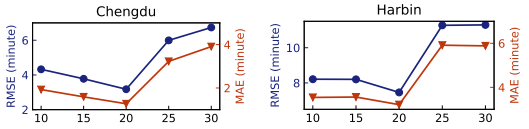
在表 5-1 中列出的最优超参数是在验证集上进行参数实验时选择的。本实验评估这些关键超参数在测试集上的效果，结果展示在图 5-10 中。从结果中能够得出的结论如下。

1）如图 5-10(a) 所示，将格网尺寸 L_G 设置为 20 是最优的。更大的格网尺寸会导致粗粒度的 PiT，这对于准确的轨迹行程表示来说是不足够的。另一方面，更小的格网尺寸会增加 PiT 的稀疏性，导致模型对微小的轨迹差异更为敏感。有趣的是，增加格网尺寸对预测性能的影响要比减小格网尺寸更大。这变相证明了将轨迹行程建模为 PiT 会减少噪音对轨迹行程信息挖掘的干扰。

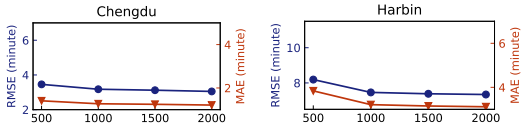
2）图 5-10(b) 表明更多的扩散步骤会产生更好的结果。直观地说，可以通过更多的步骤更好地学习噪声。然而，当 N 超过 1000 时，收益会减小。因此，本章最终选择 $N = 1000$ 作为准确率和效率之间的平衡。

3）L_D、d_E 和 L_E 决定了 PiT 推断模型和旅行时间估计的模型表达能力。图 5-

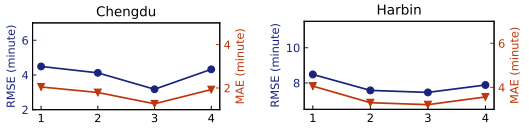
10(c)、5-10(d) 和 5-10(e) 展示了它们的有效性。所有这些参数都有最优值。模型太小会导致难以提取数据集中的复杂信息，而模型太大则会导致过拟合。



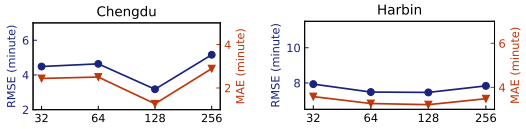
(a) 网格长度 L_G 的影响



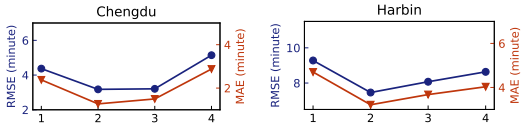
(b) 扩散总步数 N 的影响



(c) Unet 层数 L_D 的影响



(d) Transformer 嵌入维度 d_E 的影响



(e) Transformer 层数 L_E 的影响

图 5-10　超参数对起终点旅行时间估计性能的影响

Figure 5-10 Effects of hyper-parameters on OD ETA performance.

(3) 消融研究

为了评估所提出方法中特征和模块的有效性，本实验围绕如下几个 IGOP 的变体进行了消融研究。

- **Routing+Est.**：将基于路由的基线方法与 PiT 旅行时间估计相结合。
- **Infer.+Path-based**：将 IGOP 推断的轨迹行程与基于路径的基线方法相结合。
- **No-t**：从 ODT 输入中删除出发时间 t_o。
- **No-od**：从 ODT 输入中删除出发和目的地坐标。
- **No-odt**：完全删除条件信息 ODT 输入。
- **No-CE**：从 PiT 旅行时间估计中删除单元格嵌入模块。
- **No-ST**：从 PiT 旅行时间估计中删除全连接映射模块。
- **Est-CNN**：将 MViT 替换为基于 CNN 的估计器。
- **Est-ViT**：将 MViT 替换为常规的 ViT。

表 5-6　特征和模块对起终点旅行时间估计性能的影响
Table 5-6 Effects of features and modules on OD ETA performance.

数据集	成都 / 哈尔滨		
指标	RMSE (分钟)	MAE (分钟)	MAPE (%)
Dijkstra+Est.	9.182/11.869	6.871/8.246	41.462/50.488
DeepST+Est.	4.587/8.879	3.170/5.689	23.437/33.769
Infer.+WDDRA	3.773/7.958	1.801/4.171	18.937/31.514
Infer.+STDGCN	3.476/7.611	1.664/3.818	17.653/29.756
No-t	4.325/8.798	1.926/4.345	16.820/35.973
No-od	7.355/10.947	4.564/6.333	38.879/51.699
No-odt	8.466/11.172	5.880/6.562	49.830/53.331
No-CE	3.778/8.584	1.591/4.144	14.034/34.441
No-ST	7.784/11.023	5.036/6.427	42.850/52.442
Est-CNN	6.297/10.389	3.500/5.765	30.004/47.166
Est-ViT	3.229/**7.390**	1.293/**3.187**	11.547/**26.484**
IGOP	**3.177**/7.462	**1.272**/3.213	**11.343**/26.698

粗体表示最佳结果，下划线 表示次好结果。

将这些变体的性能与完整的 IGOP 模型和 PiT 旅行时间估计模型进行比较，结果如表 5-6 所示。有以下观察：

1）将路径规划方法与提出的 PiT 旅行时间估计阶段结合，未能产生令人满意的结果。这主要是因为这些方法推断出的路线不准确。另外，这些路径规划方法未生成时态特征，即 PiT 中的第二和第三个通道，因此这些特征是基于单元格之间历史平均旅行时间进行填充的。相比之下，IGOP 基于从历史轨迹中学到的行程信息推断出 PiT 的时空特征，有助于提高旅行时间估计的准确性。

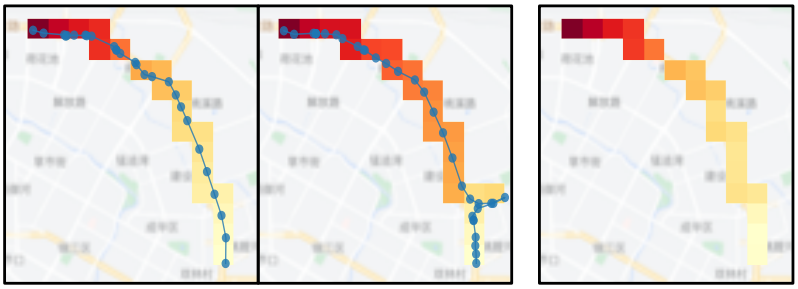
2）将 IGOP 推断的 PiT 与基于路径的方法相结合，相较于表 5-4 中的结果，提高了它们的估计性能。这些改进可以归因于与由 DeepST 产生的路径相比，IGOP 推断出的路径更准确。但是，这些结果仍然不及 PiT 旅行时间估计所获得的结果，因为 WDDRA 和 STDGCN 中使用的 RNN 模型在建模 PiT 中的时空关联方面不如所提出的 MViT 效果好，这也在其他时间序列挖掘的研究中得到了证明[73,124]。

3）删除条件 ODT 输入中的特征会降低估计性能，因为推断出的 PiT 不能准确地对应给定的 ODT 输入。

4）从 PiT 旅行时间估计中删除嵌入模块也会使预测变得更糟，因为 PiT 中的时空信息没有得到全面利用。

5）从 Est-CNN 和 IGOP 的结果比较可以证明，在处理 PiT 时，基于 Transformer 的 MViT 比基于 CNN 的方法更有效。从 Est-ViT 和 IGOP 的结果比较可以看出，MViT 的估计性能与 ViT 非常接近。结合图 5-9 中的效率比较，所提出的 MViT 可以在不损害估计性能的情况下提高效率。

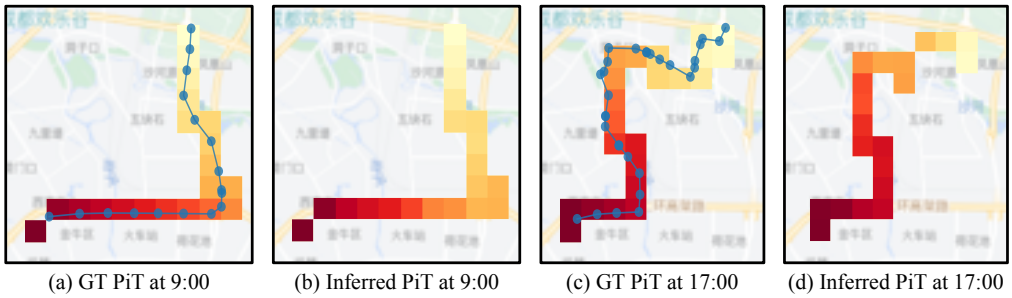
(4) 案例分析



(a) GT PiTs　　(b) Inferred PiT

图 5-11　同一起终点和出发时间段对应的轨迹

Figure 5-11 Trajectories for the same OD pair and departure at the same time of the day.



(a) GT PiT at 9:00　(b) Inferred PiT at 9:00　(c) GT PiT at 17:00　(d) Inferred PiT at 17:00

图 5-12　同一起终点、不同出发时间段对应的轨迹

Figure 5-12 Trajectories for the same OD pair that depart at different times of the day.

本节在成都数据集上进行了两方面的案例研究。

第一个案例研究对特定情况下 PiT 的第三个通道（时间偏移）的推断结果和真实值进行可视化比较。该案例分为如下两种情况。

1）同一时间段出发的轨迹：从图 5-11(a) 中可以观察到，从同一时间段内出发时，两个真实 PiT 高度相似，但具有微小的差异，其中第二个 PiT 可以被看作是

一个异常值。图 5-11(b) 展示了给定真实 ODT 输入后，IGOP 模型推断得到的 PiT，而这个推断值能与真实值很好地匹配，且不受异常值的影响。

2）不同时间段出发的轨迹：从图 5-12 中可以推断，在一天中的不同时间段，从同一出发点到同一目的地的轨迹可能具有不同的路线选择，进而导致旅行时间具有很大的差异。IGOP 方法考虑了出发时间信息，因此可以考虑到不同出发时间带来的影响。

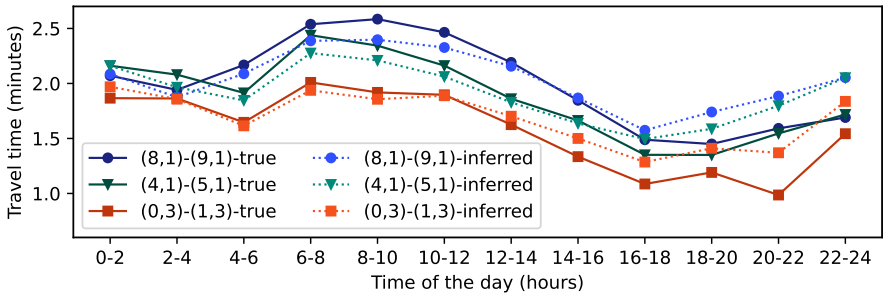


图 5-13　一天中空间网格间平均旅行时间的变化

Figure 5-13 Average travel time between spatial cells during a day.

第二个案例研究关注一天中空间网格之间的平均旅行时间的演变。可视化流程选择前 3 个最常通行的空间网格对，并将一天以两小时的间隔划分，计算每个时间间隔内空间网格间的平均旅行时间。图 5-13 展示了平均旅行时间的真实值，以及根据推断得到的 PiT 计算的值。可以看到，推断 PiT 计算得到的值与真值高度一致，这反映了推断 PiT 中的时间特征具备较高的准确率。

5.4　本章小结

本章为了解决自监督学习难以应对轨迹数据的噪音和异常值，因此无法高效地建模行程信息的挑战，进行了行程信息建模的轨迹自监督学习相关研究，并提出了一种起终点先验的轨迹行程生成模型 IGOP。该模型设计了一种对噪音鲁棒的行程表示形式，并通过一种对异常值鲁棒的自监督学习过程，建模轨迹行程与起终点的关联性。

具体而言，IGOP 提出了一种称为像素化轨迹的行程表示形式，将空间区域划分为网格，并通过多个特征通道将轨迹的原始时空特征抽象为像素特征，增强了对轨迹噪音的鲁棒性。IGOP 随后基于扩散生成提出了一种对轨迹异常值鲁棒的的自监督训练方案，建模轨迹行程与起终点的关联性，并能够学习得到起终点-行程生成模型。同时，为了能够将自监督学习模型高效地适配于起终点旅行时间估计

任务，提出了一种新颖的掩码视觉 Transformer 预测模块，通过高效的掩码实现方案，极大降低起终点旅行时间估计任务的计算开销。

IGOP 在两个出租车定位数据集上进行验证，其自监督学习得到的模型被适配于起终点行程生成与起终点旅行时间估计任务。与基线模型的对比表明 IGOP 所学习的模型能够生成更准确的行程，也能提供更精准的旅行时间估计，证明了 IGOP 自监督学习的有效性。此外，对异常值影响的相关分析证明了 IGOP 对异常值的鲁棒性，案例分析直观地展示了 IGOP 所生成行程的准确性。

6 多视图融合的轨迹自监督学习

本章旨在构建有效融合多方面信息的时空轨迹自监督学习方法，提出一种轨迹多视图最大熵嵌入模型 MMTEC。MMTEC 将时空轨迹中的多方面信息归总为离散、连续时空两个视图，并建立了一种轨迹的多视图最大熵编码自监督框架，能够有效地融合轨迹的两个视图。MMTEC 模型的自监督学习有效性在两个轨迹数据集和三种下游任务上验证。

6.1 本章引言

本研究在第 3 至 5 章提出了挖掘轨迹数据中多方面信息的时空轨迹自监督学习方法。如第 1.2.1 节所述，在全面挖掘时空轨迹多方面信息的同时，还需要有效融合这些信息，使得构建的时空轨迹自监督学习方法适用于多种下游任务，包括未来轨迹预测[1–3]、异常检测[8–10]、轨迹聚类[8,11–13] 等。

然而，有效地从时空轨迹中建模并融合多方面信息具有一定挑战。从一方面来说，轨迹通常为长序列，这给序列模型带来了效率上的挑战。此外，轨迹固有的采样不规则性增加了从时空轨迹挖掘特征的难度。当前的轨迹自监督学习方法通常从自然语言处理（Natural Language Processing, NLP）和时间序列建模领域中汲取灵感。t2vec[51]、traj2vec[11]、GM-VSAE[8] 和 TremBR[50] 均基于循环神经网络（Recurrent Neural Network, RNN）[72] 构建轨迹编码器。此外，Toast[125] 使用了 Transformer 编码器[73]。然而，基于 RNN 的轨迹编码器会受到梯度消失问题的影响[126]，而基于 Transformer 的轨迹编码器具有二次计算复杂度，这使它们不适合用于捕捉轨迹中的长序列相关性。同时，无论是 RNN 还是 Transformer 编码器，仅在接收到新输入时更新其隐藏状态，限制了它们建模轨迹连续时空视图的能力[92]。

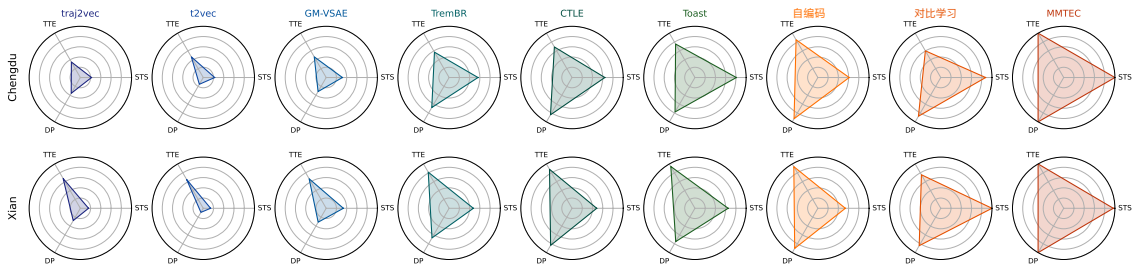


图 6-1 轨迹自监督学习模型在两个数据集和三种下游任务上的相对性能比较图

Figure 6-1 Comparison of the relative performance of trajectory embeddings on two datasets and three downstream tasks.

另一方面，现有工作并未充分考虑和融合轨迹的多方面信息。现有的自监督学习方法主要关注某一方面的信息而忽略另一方面，导致它们建模的信息并不全面。具体而言，t2vec[51]、traj2vec[11] 和 GM-VSAE[8] 优先考虑建模轨迹中的经纬度特征，但忽视了地图匹配轨迹中的路网信息。另一方面，TremBR[50] 和 Toast[125] 通过建模路网约束下的轨迹来融入旅行语义视图，但未强调对经纬度特征的建模。同时，现有工作使用的自编码[48] 和对比学习[31,125] 框架难以有效地融合多方面的信息。这些不足将导致自监督学习模型在多种下游任务上难以取得较好的性能，如图 6-1 所示。

为了解决上述挑战，本章提出一种轨迹自监督学习方法，称为**轨迹多视图最大熵**（*M*aximize *M*ulti-view *T*rajectory *E*ntropy *C*oding, **MMTEC**）**嵌入模型**，它能全面地考虑轨迹的多方面信息，并将它们有效地融合为轨迹的嵌入向量。

从技术角度，MMTEC 通过自监督学习，得到了一个输入对象为第 2.1.3 节介绍的轨迹的嵌入映射模型。该模型将时空轨迹序列映射为固定维度的嵌入向量，作为对轨迹多视图信息的融合。此类模型形式化定义如下。

定义 6.1 (轨迹嵌入映射模型, Trajectory Embedding Model). 一个轨迹嵌入映射模型 f_θ 将给定的时空轨迹 $\mathcal{T}$ 映射为其对应的嵌入向量 $\boldsymbol{e}_\mathcal{T} \in \mathbb{R}^d$，即 $f_\theta(\mathcal{T}) = \boldsymbol{e}_\mathcal{T} \in \mathbb{R}^d$，其中 d 是表示向量的维度。

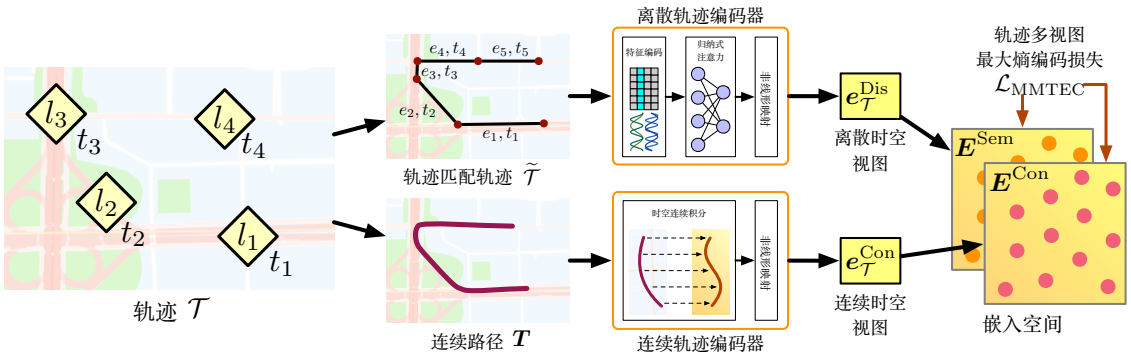


图 6-2　MMTEC 嵌入模型的整体框架

Figure 6-2 The overall framework of the MMTEC embedding model.

本研究内容的主要工作总结如下：

1）面向多视图融合的轨迹自监督学习，提出了轨迹多视图最大熵嵌入模型，将轨迹的多方面信息建模为离散、连续时空两个视图，并有效融合两个视图为轨迹的嵌入向量。

2）构建了基于注意力机制的离散轨迹编码器和基于 NeuralCDE 的连续轨迹编码器，能够有效且高效地将轨迹建模为离散、连续时空视图。

3）提出了轨迹的多视图最大熵自监督学习框架，能够有效地融合两方面视图，并学习轨迹的嵌入映射模型。

4）在两个轨迹数据集和三个下游任务上进行了一系列实验,结果表明 MMTEC 具有良好的泛化性能。

6.2 轨迹多视图最大熵嵌入模型

6.2.1 模型整体结构

图 6-2 展示了本章所提出的 MMTEC 框架。给定一条时空轨迹，目标是将轨迹的多方面信息建模为离散、连续时空两方面视图，并将它们有效融合为轨迹的嵌入向量。为了有效地建模并融合轨迹的两个视图，MMTEC 提出了一种时空轨迹的多视图最大熵自监督学习方案。

具体而言，MMTEC 将轨迹进行地图匹配，并使用基于注意力机制的离散式轨迹编码器，高效地从地图匹配轨迹中提取离散时空视图。同时，MMTEC 恢复轨迹的连续时空关联性，并使用基于 NeuralODE 的连续轨迹编码器对连续时空视图进行建模。MMTEC 通过定义视图一致性先验约束，将这两方面视图融入自监督训练任务中。自监督训练后，这两种嵌入被组合在一起用于下游任务。

6.2.2 轨迹的最大熵编码

当前的轨迹自监督学习方法通常在训练任务中使用自编码式或对比式损失，不能有效地融合轨迹的多个视图，限制所学表示的通用性进而使其不能适用于不同类型的下游任务。为了解决这个问题，MMTEC 从第 2.3.2 节中介绍的最大熵编码（MEC）中获得灵感，并开发了一种自监督训练任务来最小化训练过程中引入的偏差。

假设有一组轨迹 $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ 以及它们相应的表示向量 $\boldsymbol{E}_{\mathcal{T}} = \{\boldsymbol{e}_{\mathcal{T}_1}, \boldsymbol{e}_{\mathcal{T}_2}, \ldots, \boldsymbol{e}_{\mathcal{T}_N}\}$，其中 $\boldsymbol{e}_{\mathcal{T}_i} \in \mathbb{R}^d$。为了最小化自监督学习中引入的偏差，基于 MEC 框架来最大化表示向量的熵，即使用公式 (2-7) 的负值作为自监督训练目标函数。使用泰勒展开法简化公式 (2-7)，并遵照 MEC[80] 的做法得到更容易计算的目标函数：

$$\mathcal{L} = -\text{trace}\left(\frac{N+d}{2}\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{k}\left(\frac{d}{N\varepsilon^2}\boldsymbol{E}_{\mathcal{T}}^{\top}\boldsymbol{E}_{\mathcal{T}}\right)^k\right), \tag{6-1}$$

由此训练的表示向量集 $\boldsymbol{E}_{\mathcal{T}}$ 具有最大的熵值，因此应当能在各种下游任务中获得较为一致的性能。

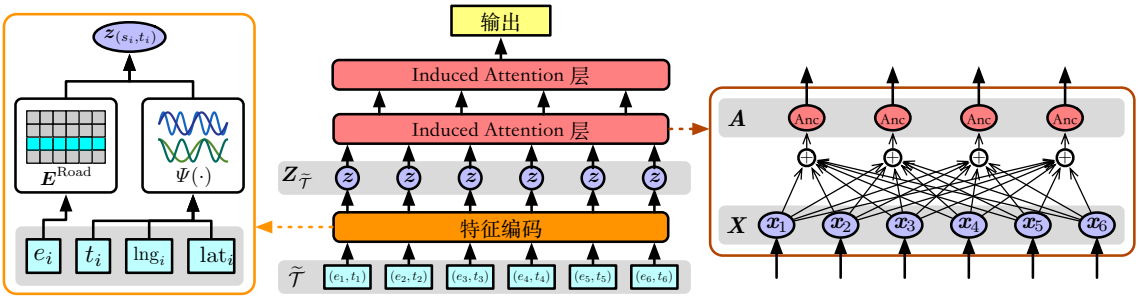
需要注意的是，最大熵原理要求给出一个明确定义的先验条件。如果仅仅使用公式 (6-1) 对轨迹表示模型进行预训练，而缺乏有效的先验条件定义，将导致所有轨迹都被表示为均匀分布[79]。此外，仅训练一种形式的轨迹表示会限制自监督学习的全面性。例如，仅表示地图匹配的轨迹，则捕捉了离散时空视图而忽略了轨迹的连续时空视图。

基于这些考虑，MMTEC 提出了一种多视图轨迹方案，将轨迹的多方面信息建模为两个视图。随后，MMTEC 将多视图一致性作为自监督学习目标的先验条件，将两个视图有效融合为更全面的轨迹嵌入向量。

6.2.3 轨迹多视图建模

本节将时空轨迹中的多方面信息归总为离散时空与连续时空两个视图，并分别提出专门设计的轨迹编码器，有效地将轨迹建模为两个视图。

(1) 离散时空视图建模



(a) 编码路段特征　(b) 离散轨迹编码器　(c) 归纳式注意力层

图 6-3　离散轨迹编码器的结构和其组件

Figure 6-3 The architecture of the discrete trajectory encoder and its components.

时空轨迹数据由离散采样的轨迹点构成，其中每个轨迹点记录了对象在某一时刻所处的特定空间地点。MMTEC 将离散轨迹点中包含的信息归总为离散时空视图。为了从轨迹中建模该视图，MMTEC 首先将轨迹映射到路网上，以更准确地描述离散轨迹点中包含的信息。换言之，地图匹配轨迹 $\widetilde{\mathcal{T}}$ 可以看作原始轨迹 $\mathcal{T}$ 的离散时空特征。然而，$\widetilde{\mathcal{T}}$ 并不适合直接作为 $\mathcal{T}$ 的离散时空视图，因为它既包含离散特征（即路段索引），又包含数值特征（即时间戳和路段坐标），且是不等长序列。

为了解决这个问题，本节提出了一个基于注意力的离散编码器，它包含特征编码层和归纳式注意力层。这个编码器可以有效地挖掘 $\widetilde{\mathcal{T}}$ 中的路段间的关联，并

将 $\widetilde{\mathcal{T}}$ 映射成固定长度的表示向量，如图 6-3 所示。该编码器的实现如下。

特征编码层： $\widetilde{\mathcal{T}}$ 中的每个轨迹点都包含一些特征：路段索引 e_i，时间戳 t_i，以及空间坐标 $(\mathrm{lng}_i, \mathrm{lat}_i)$。

一个索引提取嵌入模块用于嵌入离散的路段索引 e_i，以捕获路段的内在语义信息。具体地，初始化一个嵌入矩阵 $\boldsymbol{E}^{\mathrm{Road}} \in \mathbb{R}^{d\times|\mathcal{E}|}$，其中每一列 $\boldsymbol{E}^{\mathrm{Road}}_{e_i}$ 对应于路段 e_i 的嵌入向量。

对于连续值特征 $t_i, \mathrm{lng}_i, \mathrm{lat}_i$，使用三角函数 $\boldsymbol{\Psi}: \mathbb{R} \to \mathbb{R}^d$ 进行编码，以突出它们的周期性特征，例如在同一天的相同时间段内的出行行为相似性。该函数受到了 Transformer[73] 中使用的位置编码的启发。编码函数 $\boldsymbol{\Psi}(v)$ 具体计算如下：

$$\boldsymbol{\Psi}(v) = (\cos(\omega_1 v), \sin(\omega_1 v), \ldots, \cos(\omega_{d/2} v), \sin(\omega_{d/2} v)), \tag{6-2}$$

其中 $v \in \{t_i\} \cup \{\mathrm{lng}_i\} \cup \{\mathrm{lat}_i\}, i \in \{1, 2, \ldots, |\widetilde{\mathcal{T}}|\}$ 表示输入的数值特征，$\omega_1, \omega_2, \ldots, \omega_{d/2}$ 是可学习的参数。需要注意的是，对于不同类型的输入，这些参数不共享。借助 $\boldsymbol{\Psi}$，编码器能够利用 sin 和 cos 函数捕获数值特征的时空周期性特征。此外，它还有助于模型更好地理解不同轨迹点之间的相对关系。这是因为两条编码向量 $\boldsymbol{\Psi}(v)$ 和 $\boldsymbol{\Psi}(v+\delta)$ 的点积计算如下：

$$\boldsymbol{\Psi}(v) \cdot \boldsymbol{\Psi}(v+\delta) = \cos(\omega_1 \delta) + \cos(\omega_2 \delta) + \cdots + \cos(\omega_{d_v/2} \delta), \tag{6-3}$$

这意味着 $\boldsymbol{\Psi}(v)$ 和 $\boldsymbol{\Psi}(v+\delta)$ 之间的距离（由点积测量）仅取决于它们的数值差异 δ，而不取决于偏移 v。后续的轨迹编码器能够捕获这种距离信息，并建模 $\widetilde{\mathcal{T}}$ 中轨迹点之间的相关性。

最后，轨迹点 (s_i, t_i) 的潜在嵌入计算为：

$$\boldsymbol{z}_{(s_i,t_i)} = \boldsymbol{E}^{\mathrm{Road}}_{s_i} + \boldsymbol{\Psi}(t_i) + \boldsymbol{\Psi}(\mathrm{lng}_i) + \boldsymbol{\Psi}(\mathrm{lat}_i) \tag{6-4}$$

由此，可以将路网匹配轨迹 $\widetilde{\mathcal{T}}$ 转换为潜在嵌入序列：

$$\boldsymbol{Z}_{\widetilde{\mathcal{T}}} = \langle \boldsymbol{z}_{(s_1,t_1)}, \boldsymbol{z}_{(s_2,t_2)}, \ldots, \boldsymbol{z}_{(s_{|\widetilde{\mathcal{T}}|}, t_{|\widetilde{\mathcal{T}}|})} \rangle \in \mathbb{R}^{d\times|\widetilde{\mathcal{T}}|} \tag{6-5}$$

归纳式注意力编码器： 为了从潜在嵌入序列 $\boldsymbol{Z}_{\widetilde{\mathcal{T}}}$ 建模捕捉路段间关联，一种直观的做法是使用 Transformer 编码器。然而，Transformer 中的自注意机制具有二次时间和内存复杂度，这意味着它在处理长序列时的可扩展性欠佳。因此，本节从 NLP 中顺序模型相关研究中获得灵感。具体而言，本节使用了一种能够保证计算复杂度较低的前提下，考虑长序列关联性的注意机制。该机制称为归纳式注意力机制，利用全局共享的锚点序列来计算注意力权重。

对于每层归纳式注意力层，初始化一条锚点序列 $\boldsymbol{A}_i \in \mathbb{R}^{d \times L_{A_i}}$，作为该层的固有参数，其中 i 为层序号，L_{A_i} 是锚点序列的长度。归纳式注意力层 IA_i 通过注意力机制和前馈网络实现：

$$\begin{aligned} \mathrm{IA}_i(\boldsymbol{X}_i) &= \mathrm{Norm}(\boldsymbol{H}_i + \mathrm{FFN}_i(\boldsymbol{H}_i)) \\ \boldsymbol{H}_i &= \mathrm{Norm}(\boldsymbol{A}_i + \mathrm{Att}_i(\boldsymbol{A}_i, \boldsymbol{X}_i, \boldsymbol{X}_i)), \end{aligned} \tag{6-6}$$

其中，Norm、FFN_i 和 Att_i 分别表示层归一化[127]、前馈网络和第 i 层的多头点积注意力，其中注意力头数为 8。$\boldsymbol{X}_i \in \mathbb{R}^{d \times L_{x_i}}$ 是第 i 层的输入。在注意力计算过程中，$\boldsymbol{A}_i$ 被视为查询（query），而 $\boldsymbol{X}_i$ 被视为键（key）和值（value）。FFN_i、Att_i 以及锚点序列 $\boldsymbol{A}_i$ 的参数在小批量（mini-batch）之间是共享的，但在不同层之间是不共享的。

离散编码器 DisEnc 通过堆叠两个 IA 层来构建。给定一条潜在嵌入序列 $\boldsymbol{Z}_{\widetilde{\mathcal{T}}}$，DisEnc 的定义如下：

$$\mathrm{DisEnc}(\widetilde{\mathcal{T}}) = \mathrm{IA}_2(\mathrm{IA}_1(\boldsymbol{Z}_{\widetilde{\mathcal{T}}})), \tag{6-7}$$

其中第二层的锚点序列长度 L_{A_2} 被设置为 1，以保证 DisEnc 的输出维度为 d。

通过在注意力权重计算中共享锚点序列，而不是采用自注意力，本节构建了一个更高效的离散编码器，能够高效地建模路网匹配轨迹中的长序列相关性。

最终的离散时空视图通过在 DisEnc 的输出上附加非线性映射得到：

$$\boldsymbol{e}_{\mathcal{T}}^{\mathrm{Dis}} = \boldsymbol{W}_2(\sigma(\boldsymbol{W}_1 \mathrm{DisEnc}(\widetilde{\mathcal{T}}))), \tag{6-8}$$

其中 $\boldsymbol{e}_{\mathcal{T}}^{\mathrm{Dis}} \in \mathbb{R}^d$ 代表了轨迹的离散时空视图。$\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{d \times d}$ 是映射矩阵，σ 表示非线性激活函数，这里使用 ReLU 激活函数。

(2) 连续时空视图建模

时空轨迹背后对应的是对象在空间中的连续运动行为，可以将这种行为内蕴含的信息归总为连续时空视图。然而，轨迹点的不规则和离散特性，给传统的序列模型如 RNN[72] 和 Transformer[73] 建模连续时空视图带来了挑战。因此，本节采用一种可以建模轨迹隐含的连续路径的方法，如图 6-4 所示。

首先，根据轨迹的 GPS 点恢复其连续路径。给定轨迹 $\mathcal{T} = \langle (l_1, t_1), (l_2, t_2), \ldots, (l_{|\mathcal{T}|}, t_{|\mathcal{T}|}) \rangle$，通过计算其三次样条插值（cubic spline）来拟合轨迹对应的连续路径：

$$\boldsymbol{T} = \mathrm{Spline}(\mathcal{T}), \tag{6-9}$$

其中 Spline 是三次埃尔米特样条（cubic Hermite spline），其在节点 $t_1, t_2, \ldots, t_{|\mathcal{T}|}$ 处的值分别为：

$$\boldsymbol{T} : [t_1, t_{|\mathcal{T}|}] \to \mathbb{R}^2,\ \boldsymbol{T}_{t_i} = l_i,\ i \in \{1, 2, \ldots, |\mathcal{T}|\} \tag{6-10}$$

由于 $\boldsymbol{T}$ 拟合了轨迹 $\mathcal{T}$ 隐含的连续路径，它可以作为将 $\mathcal{T}$ 映射到连续时空表示的中间值。由于 $\boldsymbol{T}$ 的连续性定义，将它直接应用于大多数轨迹分析任务是不合适的。因此，本节提出了一个连续编码器 ConEnc，可以建模由 $\boldsymbol{T}$ 表示的连续路径。

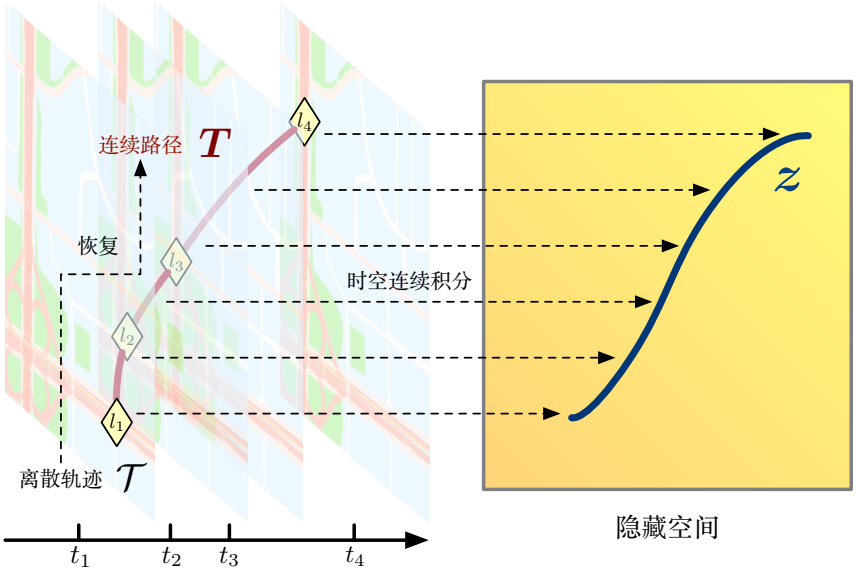


图 6-4　连续轨迹编码器的图示

Figure 6-4 Illustration of the continuous trajectory encoder.

为了构建连续编码器 ConEnc，本节采用常微分方程的一个分支，即神经控制的微分方程（Neural Controlled Differential Equation，NeuralCDE）[128]。该方法可以捕捉连续空间中的时序过程，并以连续方式建模输入特征和隐藏状态。基于现有的 NeuralCDE 公式[128]，本节所提出的连续编码器定义如下：

$$\begin{aligned} \text{ConEnc}(\boldsymbol{T}) &= \boldsymbol{z}_{t_1} + \int_{t_1}^{t_{|\mathcal{T}|}} \text{CDENet}(\boldsymbol{z}_t)\mathrm{d}\boldsymbol{T}_t \\ &= \boldsymbol{z}_{t_1} + \int_{t_1}^{t_{|\mathcal{T}|}} \text{CDENet}(\boldsymbol{z}_t)\frac{\mathrm{d}\boldsymbol{T}}{\mathrm{d}t}(t)\mathrm{d}t, \end{aligned} \tag{6-11}$$

其中 $\boldsymbol{z}_{t_1} = \text{InitNet}(l_1, t_1)$ 是一个参数化的初始状态，$\text{InitNet} : \mathbb{R}^3 \to \mathbb{R}^d$ 表示用于计算初始状态的神经网络，$\text{CDENet} : \mathbb{R}^d \to \mathbb{R}^{d\times 3}$ 表示 NeuralCDE 的积分核。InitNet 和 CDENet 的实现如下。

参数化的初始状态和积分核： 初始状态 $\boldsymbol{z}_{t_1}$ 确定了 NeuralCDE 积分的起始点，并且依赖于轨迹的第一个点 (l_1, t_1)。由于轨迹点中的所有特征都是数值型的，这里

使用公式 (6-2) 中的三角函数来计算初始状态。因此，神经网络 InitNet 定义为：

$$\text{InitNet}(l_1, t_1) = \Psi(\text{lng}_1) + \Psi(\text{lat}_1) + \Psi(t_1) \tag{6-12}$$

神经网络 CDENet 用作 NeuralCDE 的积分核。其实现为两层的非线性映射：

$$\text{CDENet}(\boldsymbol{z}_t) = \boldsymbol{W}_6(\sigma(\boldsymbol{W}_5\boldsymbol{z}_t + \boldsymbol{b}_5)) + \boldsymbol{b}_6, \tag{6-13}$$

其中 $\boldsymbol{W}_5 \in \mathbb{R}^{3d\times d}$ 和 $\boldsymbol{W}_6 \in \mathbb{R}^{3d\times 3d}$ 是映射矩阵，$\boldsymbol{b}_5, \boldsymbol{b}_6 \in \mathbb{R}^{3d}$ 是偏置项。

与公式 (6-8) 类似，本节将非线性映射附加在连续轨迹编码器上，得到最终的连续时空视图：

$$\boldsymbol{e}_{\mathcal{T}}^{\text{Con}} = \boldsymbol{W}_4(\sigma(\boldsymbol{W}_3\text{ConEnc}(\boldsymbol{T}))), \tag{6-14}$$

其中，$\boldsymbol{e}_{\mathcal{T}}^{\text{Con}}$ 代表轨迹 $\mathcal{T}$ 的连续时空视图，而 $\boldsymbol{W}_3, \boldsymbol{W}_4 \in \mathbb{R}^{d\times d}$ 是映射矩阵。

(3) 双视图融合

轨迹 $\mathcal{T}$ 的最终表示向量通过将旅行语义视图和连续时空视图组合得到：

$$\boldsymbol{e}_{\mathcal{T}} = \boldsymbol{e}_{\mathcal{T}}^{\text{Dis}} \oplus \boldsymbol{e}_{\mathcal{T}}^{\text{Con}}, \tag{6-15}$$

其中，$\oplus$ 表示沿特征维度的连接操作。该轨迹表示包含轨迹中的全面信息，旨在提升多种下游任务的性能。

6.2.4 最大熵自监督训练

为了有效地融合公式 (6-8) 和公式 (6-14) 中建模的离散、连续时空视图，本节基于视图一致性原则，引入可验证的先验信息到公式 (6-1) 中。

在梯度下降训练过程中，给定某一轨迹批次 $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$，其中 N 为批次大小。首先计算它们对应的旅行语义视图 $\boldsymbol{E}^{\text{Dis}} = \{\boldsymbol{e}_{\mathcal{T}_1}^{\text{Dis}}, \boldsymbol{e}_{\mathcal{T}_2}^{\text{Dis}}, \ldots, \boldsymbol{e}_{\mathcal{T}_N}^{\text{Dis}}\} \in \mathbb{R}^{d\times N}$，和连续时空视图 $\boldsymbol{E}^{\text{Con}} = \{\boldsymbol{e}_{\mathcal{T}_1}^{\text{Con}}, \boldsymbol{e}_{\mathcal{T}_2}^{\text{Con}}, \ldots, \boldsymbol{e}_{\mathcal{T}_N}^{\text{Con}}\} \in \mathbb{R}^{d\times N}$。

将 $\boldsymbol{E}^{\text{Dis}}$ 和 $\boldsymbol{E}^{\text{Con}}$ 之间的视图一致性先验融入到公式 (6-1) 中，并通过截断泰勒展开来使得计算可行，可以得到 MMTEC 自监督学习的目标函数：

$$\mathcal{L}_{\text{MMTEC}} = -\text{trace}\left(\frac{N+d}{2}\sum_{k=1}^{K}\frac{(-1)^{k+1}}{k}\left(\frac{d}{N\varepsilon^2}\boldsymbol{E}^{\text{Dis}\top}\boldsymbol{E}^{\text{Con}}\right)^k\right) \tag{6-16}$$

其中，K 是泰勒展开阶数，作为超参数调整。该目标函数在最大化信息熵的同时保证视图一致性，能够经过自监督训练有效融合时空轨迹的两个视图，得到到通用且全面的轨迹表示模型。

6.3 实验

为了评估所提出的 MMTEC 的通用性，在两个轨迹数据集上进行实验，并将 MMTEC 应用于三种下游任务。

6.3.1 基线模型

为了评估所提出模型相对于现有方法的优势，实验中 MMTEC 与六种轨迹自监督学习方法展开比较。

- **traj2vec**[11]：通过计算连续轨迹点之间的时空特征差异构建特征序列，并应用自回归预训练任务来学习表示。
- **t2vec**[51]：基于去噪自编码器对 RNN 轨迹编码器进行预训练，旨在从稀疏轨迹中恢复稠密的轨迹信息。
- **GM-VSAE**[8]：使用 RNN 变分轨迹编码器将轨迹映射到多维高斯空间，并按照变分自编码器的方式进行预训练。
- **TremBR**[50]：构建 RNN 自编码器，同时恢复路段和时间戳来预训练轨迹表示。
- **CTLE**[129]：预训练双向 Transformer 进行两种基于掩码语言模型的任务，然后基于上下文计算地点的嵌入。
- **Toast**[125]：将 node2vec 模型应用于道路网络，预训练一组路段的嵌入，然后通过基于掩码语言模型和序列判别的任务预训练 Transformer 编码器来构建轨迹表示。

6.3.2 实验设置

MMTEC 和各基线模型在 2.4.3 节中介绍的相似轨迹搜索、轨迹预测和轨迹旅行时间估计任务上验证。具体而言，相似轨迹搜索使用余弦相似度计算轨迹嵌入向量的相似度。轨迹预测任务采用方案（2）实现，预测模块为全连接网络，将一条完整轨迹去除最后 5 个轨迹点后作为历史轨迹，预测最后一个点的路段下标。轨迹旅行时间估计的预测模块用全连接网络实现。

考虑到对离散和连续时空视图的建模在采样相对稠密的轨迹数据上更有效，实验选用 2.2.3 中介绍的成都和西安出租车定位数据集。所有数据集均按照轨迹的起始时间进行排序，并按照 8:1:1 的比例划分为训练集、评估集和测试集。预训练

嵌入方法和下游预测器都在训练集上进行训练。预训练嵌入方法训练 30 轮，而下游预测器则在评估集上实现早停技术。最终的指标是在测试集上计算得到的。

实验中使用 Top-N 准确率（即 Acc@N，$N=1,5$）来评估相似轨迹搜索任务；用 Acc@N（$N=1,5$）和 macro-F1 来评估轨迹预测任务；用平均绝对误差（MAE）、均方根误差（RMSE）和平均绝对百分比误差（MAPE）来评估旅行时间估计任务。每组实验均运行 10 次，并报告指标的平均值和标准差。

所提出的方法包含五个关键超参数，其范围和最优值列在表 6-1 中。最优超参数的选择基于成都数据集的验证集上进行的相似轨迹搜索任务的 Acc@1 结果。第 6.3.4 节中将进一步展示这些超参数的具体效果。所有模型均使用 PyTorch[102] 实现。

表 6-1　超参数的选择范围和最佳值

Table 6-1 Hyper-parameter ranges and optimal values.

参数	选择范围
d	32, 64, 128, 192, 256
N	128, 256, 384, 512, 640, 768, 896, 1024
ε^2	128, 256, 384, 512, 640, 768, 896, 1024
L_{A_1}	0, 4, 8, 12, 16, 20, 24, 28, 32
K	1, 2, 3, 4, 5, 6, 7, 8, 9, 10

下划线 示意最佳值。

6.3.3　总体性能对比

表 6-2 和 6-3 综合比较了不同轨迹表示学习方法在三种任务中的性能。图 6-1 中也提供了结果的可视化展示。所提出的 MMTEC 方法在各项任务中均表现优越，这证明它在生成通用和全面的轨迹嵌入方面是表现最好的。

预训练任务的通用性： traj2vec、t2vec、GM-VSAE 和 TremBR 都采用了自编码或自回归框架。然而，自回归预训练方法主要注重恢复轨迹的原始细节特征，使得学习到的表示偏向于点级的下游任务，如旅行时间估计和轨迹预测。正如结果所示，自回归预训练的嵌入方法在序列级任务，如相似轨迹搜索中表现不佳。

CTLE 和 Toast 使用了 BERT[29] 中的掩码语言模型（MLM）预训练任务。与自回归任务相比，MLM 利用了 Transformer 的双向性，有助于模型更好地理解轨迹点之间的相关性。然而，作为一种生成式的预训练任务，它依然会使学习到的嵌

表 6-2　在成都数据集上的轨迹表示学习方法性能比较

Table 6-2 Performance comparison of trajectory representation learning methods on the Chengdu dataset.

下游任务	相似轨迹搜索		旅行时间估计			轨迹预测		
指标	Acc@1(%)	Acc@5(%)	MAE(分钟)	RMSE(分钟)	MAPE(%)	Acc@1(%)	Acc@5(%)	F1(%)
traj2vec	61.38±0.78	71.19±0.59	2.570±0.022	3.740±0.020	35.11±0.24	45.54±0.27	68.23±0.20	17.83±0.09
t2vec	61.52±1.14	71.70±1.56	2.518±0.024	3.695±0.031	33.49±0.30	39.52±0.13	63.36±0.40	13.51±0.12
GM-VSAE	64.28±0.87	76.17±0.49	2.515±0.033	3.732±0.041	33.49±0.32	44.37±0.63	67.30±0.86	17.64±0.11
TremBR	71.51±2.51	79.49±1.65	2.445±0.043	3.633±0.063	32.01±0.45	55.04±0.55	77.74±0.26	26.89±0.13
CTLE	73.75±1.41	82.69±1.62	2.341±0.035	3.558±0.071	30.37±0.51	59.88±0.26	79.92±0.13	27.66±0.13
Toast	78.67±0.71	86.40±0.60	2.320±0.028	3.477±0.069	29.44±0.44	57.99±0.64	77.14±0.85	26.17±0.99
自编码	72.89±0.84	83.20±1.21	2.239±0.042	3.341±0.033	28.10±0.93	62.52±0.12	89.26±0.14	30.04±0.14
对比学习	80.62±1.99	93.13±1.41	2.419±0.124	3.590±0.184	31.59±1.79	60.84±0.31	87.66±0.13	27.35±0.28
MMTEC	**84.27±3.02**	**95.15±1.63**	**2.079±0.109**	**3.104±0.194**	**26.02±1.00**	**64.61±0.13**	**91.71±0.06**	**33.35±0.11**

粗体表示最佳结果，下划线 表示次好结果。

表 6-3　在西安数据集上的轨迹表示学习方法性能比较

Table 6-3 Performance comparison of trajectory representation learning methods on the Xian dataset.

下游任务	相似轨迹搜索		旅行时间估计			轨迹预测		
指标	Acc@1(%)	Acc@5(%)	MAE(分钟)	RMSE(分钟)	MAPE(%)	Acc@1(%)	Acc@5(%)	F1(%)
traj2vec	58.57±0.72	77.03±0.62	3.142±0.050	5.159±0.048	30.64±0.79	44.04±0.43	67.20±0.21	13.56±0.49
t2vec	58.19±1.27	73.78±1.14	3.178±0.016	5.248±0.011	30.93±0.30	37.99±0.16	62.47±0.41	8.73±0.34
GM-VSAE	62.47±0.30	78.24±1.03	3.143±0.032	5.123±0.039	30.73±0.43	45.21±0.89	68.82±0.88	13.73±0.79
TremBR	65.55±1.34	79.55±1.63	3.037±0.028	5.073±0.022	28.65±0.31	56.78±0.61	79.33±0.98	22.14±0.90
CTLE	65.74±1.95	79.26±1.25	2.949±0.042	4.912±0.108	27.74±0.58	62.25±0.53	85.21±0.53	27.17±0.81
Toast	69.43±2.12	83.41±1.67	2.941±0.066	4.894±0.173	26.86±0.80	59.47±0.65	81.77±0.71	24.71±0.93
自编码	66.95±1.93	82.28±1.24	2.900±0.090	4.724±0.138	26.91±0.92	64.52±0.18	92.09±0.11	30.14±0.74
对比学习	**77.27±1.65**	89.92±1.22	3.158±0.049	5.086±0.111	29.47±1.38	62.50±0.14	87.46±0.07	26.75±0.41
MMTEC	76.57±3.05	**91.38±2.04**	**2.870±0.023**	**4.653±0.073**	**26.06±0.44**	**67.62±0.21**	**93.68±0.08**	**36.38±0.79**

粗体表示最佳结果，下划线 表示次好结果。

入偏向于特定类型的下游任务。有趣的是，Toast 还实现了一个序列级别的预训练任务，使其在相似轨迹搜索任务上表现更好。

所提出的 MMTEC 方法的预训练任务基于最大熵原理构建，旨在学习在不同类型的下游任务中都表现良好的表示。这使得 MMTEC 更适合训练通用的轨迹表示。第 6.3.4 节中进一步分析了预训练任务的有效性。

融合信息的全面性： traj2vec、t2vec 和 GM-VSAE 方法旨在基于 GPS 坐标和时间戳对轨迹的时空特征进行建模，但它们并未考虑路网中的语义信息。此外，它们的轨迹编码器基于 RNN，无法有效捕捉连续的时空相关性。

相对应的，TremBR、CTLE 和 Toast 通过将轨迹映射到路网上，并使用轨迹编码器将映射后的轨迹转换为表示向量来融合语义信息。TremBR 还通过在输入特征中包含路段的访问时间来考虑时间信息。CTLE 通过时间编码层和掩码时间预训练任务来融合相对和绝对的时间信息。Toast 通过使用 node2vec 预训练路网嵌入向量来加强对语义信息的建模。然而，它们的轨迹编码器仍然依赖于 RNN 或 Transformer，这些模型以离散的方式更新隐藏状态，难以准确地建模轨迹中的连续时空相关性。

相比之下，所提出的 MMTEC 方法同时融合了离散、连续时空两方面视图。这使得学习到的嵌入向量能够全面地捕捉轨迹的信息，从而提高下游任务的性能。

综合性能对比总结： 所提出的 MMTEC 方法利用最大熵编码和轨迹多视图方法的优势，实现了全面、有效的轨迹自监督学习方法。该方法捕捉了轨迹中的离散、连续时空视图，提供了全面的信息，从而提高了下游任务的性能。

6.3.4 模型分析

(1) 自监督训练过程的有效性

为了评估自监督训练过程的有效性，本节将完整的 MMTEC 方法与其以下两种变体进行比较：

- **no-ft**: 不执行微调。经过自监督训练后，轨迹编码器的参数被固定，在下游任务中不参与反向传播。
- **end2end**: 以端到端的方式训练模型。轨迹编码器的参数在随机初始化后，直接通过下游任务的监督信息进行训练。

图 6-5 展示了西安数据集上轨迹预测任务的平均损失值和验证准确率变化。可以看到，自监督训练-微调的方案相比端到端的方案，具备更快的收敛速度。这说

明自监督训练后的轨迹编码器具备更佳的泛化性能和更快的收敛速度，有助于其在下游任务中以更少的训练轮次达到更好的性能。即便自监督训练后不对轨迹编码器进行微调，它依然能够取得不俗的结果，表明自监督训练过程提取了独立于特定下游任务的通用性特征。

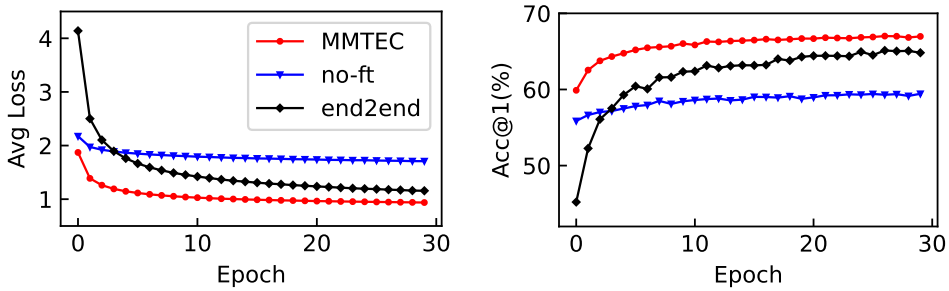


图 6-5　不同变体在训练过程中的平均损失值和准确率演化

Figure 6-5 Average loss value and validation accuracy through the training processes of different variances.

(2) 自监督学习设计的有效性

为了评估 MMTEC 方法中自监督学习设计的有效性，本节将完整的 MMTEC 方法与以下自监督学习方案进行比较：

- **自编码**：通过轨迹表示向量还原原始轨迹序列。使用两个 Transformer 解码器分别与离散和连续轨迹编码器配对，以自回归方式进行轨迹还原。两组编码器-解码器分别进行训练。
- **对比学习**：通过区分正样本和一组负样本进行自监督训练。对于每个批量中的轨迹，将其连续时空视图视为目标，其语义视图视为正样本，批量中其他轨迹的语义视图视为负样本。选择 InfoNCE[61] 作为目标函数。

它们在不同下游任务上的性能如表 6-2、 6-3 和图 6-1 所示。可以观察到，当使用自编码方案进行训练时，表示模型会偏向于点级任务；相反，当使用对比学习方案训练时，表示模型会偏向于序列级任务。而所提出的 MMTEC 方法在不同的下游任务均能取得较好的性能。

(3) 超参数的影响

这里研究表 6-1 中所列超参数对成都数据集上的测试集的影响。使用相似轨迹搜索任务的 Acc@1 和 Acc@5 指标进行验证，因为该任务不涉及微调，可以直

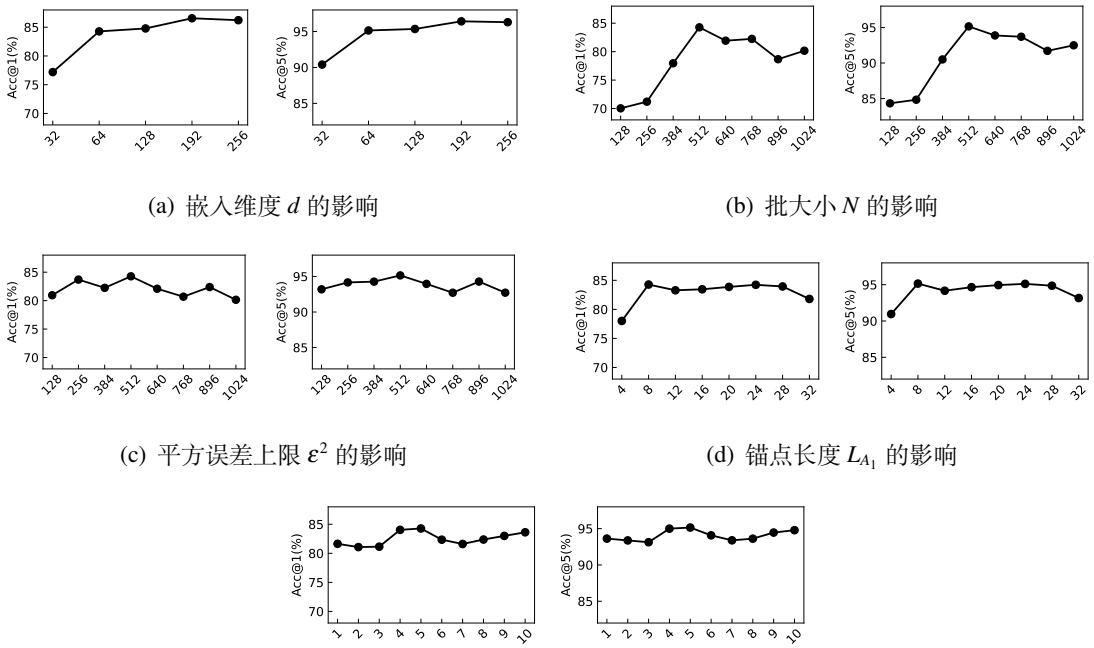
接反映表示学习模型的质量。结果如图 6-6 所示，可以得出以下观察结果：

1）如图 6-6(a) 所示，增加表示向量的维度通常会提高性能。然而，在 $d > 64$ 之后，准确性的提升几乎可以忽略不计，而计算和内存要求显著增加。因此，$d = 64$ 是性能和效率之间的平衡点。

2）批量大小 N 和平方误差上界 ε^2 都会影响预训练目标函数的系数。如图 6-6(b) 和 6-6(c) 所示，它们的最佳值均为 512。

3）离散轨迹编码器中第一 IA 层的锚点长度 L_{A_1} 对性能有影响，其最佳值为 8，如图 6-6(d) 所示。较短的长度会降低模型复杂性，使得从轨迹中提取相关性变得困难；而较长的长度会增加模型容量，导致过拟合。

4）Taylor 展开的阶数 K 决定了公式 (6-16) 估计的准确性。从图 6-6(e) 来看，$K = 5$ 是较为均衡的设置。



(a) 嵌入维度 d 的影响　(b) 批大小 N 的影响

(c) 平方误差上限 ε^2 的影响　(d) 锚点长度 L_{A_1} 的影响

(e) 泰勒展开阶 K 的影响

图 6-6　超参数对成都数据集相似轨迹搜索任务的影响

Figure 6-6 Impact of hyper-parameters for the similar trajectory search task on the Chengdu dataset.

(4) 消融研究

为了评估所提出方法的不同组成部分的有效性，比较完整版本和以下变体的性能：

- **only-sem**：在下游任务中仅使用轨迹的语义视图。
- **only-con**：在下游任务中仅使用轨迹的连续时空视图。
- **no-time**：将时间戳特征从两个轨迹编码器中移除。
- **no-spatial**：将空间坐标特征从两个轨迹编码器中移除。
- **dis-no-time**：将时间戳特征从离散轨迹编码器中移除。
- **dis-no-spatial**：将空间坐标特征从离散轨迹编码器中移除。
- **con-no-time**：将时间戳特征从连续轨迹编码器中移除。
- **con-no-spatial**：将空间坐标特征从连续轨迹编码器中移除。
- **Dis-Trans**：将离散轨迹编码器替换为 Transformer 编码器。
- **Con-Trans**：将连续轨迹编码器替换为 Transformer 编码器，其输入为轨迹序列 $\mathcal{T}$。

表 6-4　成都数据集相似轨迹搜索任务上的消融研究结果

Table 6-4 Performance comparison of different variances on the similar trajectory search task, Chengdu dataset.

指标	Acc@1(%)	Acc@5(%)
only-sem	64.84±11.53	77.93±13.62
only-con	59.36±4.67	71.23±5.32
no-time	67.38±5.28	78.14±7.50
no-spatial	53.10±0.51	64.23±1.04
dis-no-time	70.72±0.49	82.53±0.91
dis-no-spatial	72.71±2.14	83.93±2.85
con-no-time	69.69±0.57	80.24±0.83
con-no-spatial	62.79±0.59	73.56±0.91
Dis-Trans	82.86±2.57	94.36±1.39
Con-Trans	74.97±3.55	88.75±1.91
MMTEC	**84.27±3.02**	**95.15±1.63**

粗体表示最佳结果，下划线 表示次好结果。

本节在成都数据集上将这些变体在相似轨迹搜索任务中的性能进行了比较。结果如表 6-4 所示，得出以下观察结果：

1）语义视图和连续时空视图在下游任务中都是有用的。仅使用其中一种无法达到完整 MMTEC 方法的准确性。这表明这两种视图捕捉了轨迹中重要且互补的

信息。

2）空间和时间特征对于两个轨迹编码器都是至关重要的。由于两个编码器在预训练过程中同时进行训练，因此即使从一个编码器中移除时空特征也会显著影响表示学习的性能。

3）用普通 Transformer 编码器替换所提出的离散轨迹编码器会略微降低准确性。这表明所提出的编码器比 Transformer 更能有效且高效地处理地图匹配轨迹序列。此外，当用 Transformer 编码器替换连续轨迹编码器时，性能下降，因为 Transformer 无法建模连续时空相关性。

6.4 本章小结

本章为了解决自监督学习难以全面、有效地建模并融合轨迹数据中的多方面信息的挑战，进行了多视图融合的轨迹自监督学习相关研究，并提出了一种新颖的轨迹多视图最大熵嵌入模型 MMTEC。该模型提出了一种轨迹的多视图最大熵编码自监督学习框架，能够全面地建模并融合轨迹的多方面信息。

具体而言，MMTEC 首先将时空轨迹中的多方面信息归总为离散时空与连续时空两个视图。随后，MMTEC 提出了基于归纳式注意力的离散轨迹编码器，将轨迹在路网上所体现的特征建模为离散时空视图。同时，MMTEC 提出了基于 NeuralCDE 的连续轨迹编码器，能够恢复轨迹对应的连续时空变化，有效地将轨迹对应的连续移动行为建模为连续时空视图。最后，MMTEC 将代表两个视图的嵌入向量融合到轨迹的多视图最大熵编码框架中，从而构建能有效融合轨迹多视图信息的自监督学习方法。

MMTEC 在两个车辆定位数据集上进行验证，其自监督学习得到的模型被适配于轨迹预测、相似轨迹搜索、轨迹旅行时间估计三种下游任务。与基线模型的对比证明 MMTEC 所学习到的轨迹嵌入向量能够在三种下游任务上均取得最佳的性能，证明了 MMTEC 自监督学习的有效性与全面性。同时，对模块的组件分析体现了 MMTEC 中各模块的有效性。

7 面向通用性的轨迹自监督学习

本章旨在构建灵活适配稀疏或不完整轨迹输入的时空轨迹自监督学习方法，提出一种通用轨迹模型 GTM。GTM 将轨迹特征划分为三个可独立掩盖与自回归生成的特征域，使得其能灵活适配多种下游任务的不完整轨迹输入。同时，GTM 构建了从稀疏轨迹恢复对应稠密轨迹的自监督学习目标，强化了模型对稀疏轨迹的鲁棒性。GTM 的有效性在两个轨迹数据集和三种下游任务上验证。

7.1 本章引言

本论文的第 3 至 6 章研究了有效挖掘并融合时空轨迹中多方面信息的自监督学习方法。这些方法建立了将时空轨迹自监督学习模型应用于多种下游任务的信息基础，为各种下游任务提供了有用的信息。另一方面，如第 1.2.2 节所述，时空轨迹自监督学习需要解决的另一项挑战在于不同任务与场景下的稀疏或不完整轨迹输入的灵活适配。时空轨迹数据在真实世界的应用中呈现多样化，不同的应用场景和下游任务可能会有不同形式和质量的轨迹数据，特别是稀疏或不完整的轨迹。为了保证时空轨迹自监督学习模型在各种场景和任务中的通用性，需要自监督学习模型能够灵活适配多样的轨迹输入。

为了解决上述挑战，实现通用性的时空轨迹自监督学习模型，本章提出了一种通用轨迹模型（*General Trajectory Model*, **GTM**）。该模型将时空轨迹中的特征划分为时间、空间、路段三个特征域，其中每个特征域均可独立掩盖及自回归生成，因此模型能够灵活地适配多种任务的不完整轨迹输入。同时，设计了从稀疏轨迹还原对应稠密轨迹的自监督学习目标，强化了模型对稀疏轨迹的鲁棒性。模型在完成自监督训练后，可直接适配于多种下游任务。

从技术角度，GTM 通过自监督学习，得到了一个输入对象为多样轨迹形式的生成模型。该模型能够适配特定下游任务的输入轨迹形式，并生成任务所需的预测目标。

本研究内容的主要工作总结如下：

1）面向通用性的轨迹自监督学习，提出了通用轨迹模型，能够灵活适配稀疏或不完整的轨迹输入，并迁移至多种下游任务。

2）将轨迹中的特征分为三个可独立掩盖与生成的域，使得所提出的模型可以灵活地适配不同任务的不完整轨迹输入，而无需依赖于两阶段方案或重新训练。

3）精心设计的自监督学习目标，强化了模型对稀疏轨迹的鲁棒性，使模型能够从稀疏轨迹中提取稠密的时空信息。

4）在两个轨迹数据集和三种下游任务上进行了全面的实验研究，揭示了所提出模型的性能特性，并提供了证据表明该模型能够实现其设计目标。

7.2 通用轨迹模型

7.2.1 模型整体结构

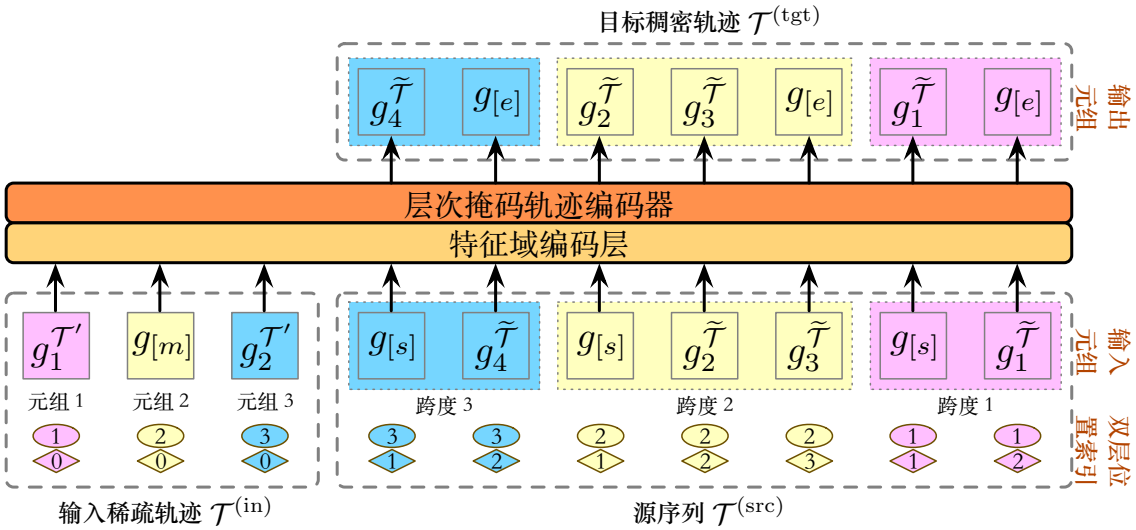


图 7-1 GTM 的整体框架图

Figure 7-1 Overall framework of the GTM.

为了构建能够应对稀疏轨迹的自监督学习模型，本章从通用语言模型（General Language Model, GLM）[130] 中汲取灵感。以图 7-2 中的轨迹为例，有稠密轨迹 $\mathcal{T}=\langle(l_1,t_1),(l_2,t_2),(l_3,t_3),(l_4,t_4)\rangle$、它的地图匹配版本 $\widetilde{\mathcal{T}}$、重新采样的稀疏轨迹 $\mathcal{T}'=\langle(l_1,t_1),(l_4,t_4)\rangle$。与 $\widetilde{\mathcal{T}}$ 相比，$\mathcal{T}'$ 缺乏时空信息和准确的路段选择。目标是自监督训练一个模型 f_θ，从 $\mathcal{T}'$ 生成 $\widetilde{\mathcal{T}}$。通过这种方式，模型建立了稀疏轨迹与其稠密、地图匹配轨迹之间的联系。

所提出模型的表示如图 7-1 所示。首先，将轨迹中的特征划分为三个域，并将每个轨迹点转换为一个特征域的元组。特征域的具体定义如下。

定义 7.1 (特征域的元组，Tuple of Feature Domains). *如图 7-2 所示，轨迹的特征被划分为三个域：1）空间域，包括与空间坐标 l_i 相关的特征；2）时间域，指时间戳 t_i；3）路段域，包括路段下标 e_i 和在路段上的行驶比例 r_i。三个域分别以蓝*

色、绿色和红色表示。可以将轨迹中的每个点转换为这三个领域的元组。例如，点 $(\tilde{l}_i, t_i)$ 被转换为 $(l_i, t_i, (e_i, r_i))$。从轨迹 $\mathcal{T}$ 中的第 i 个点转换的元组表示为 $g_i^{\mathcal{T}}$。

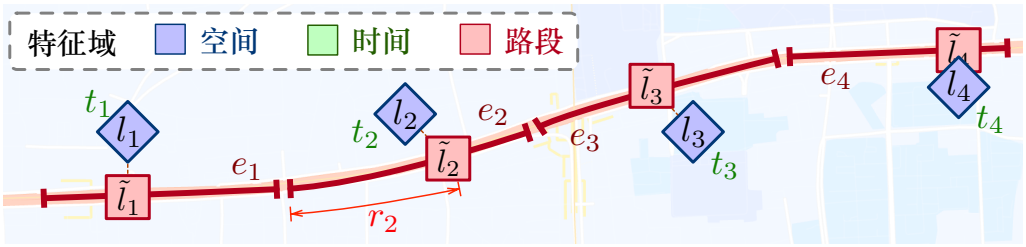


图 7-2　轨迹的三个特征域

Figure 7-2 Three feature domains of trajectories.

模型需要生成那些在 $\mathcal{T}'$ 中缺失但在 $\widetilde{\mathcal{T}}$ 中存在的特征域元组。特别地，两个元组 $g_2^{\widetilde{\mathcal{T}}}$ 和 $g_3^{\widetilde{\mathcal{T}}}$ 在 $\mathcal{T}'$ 中缺失。为了明确标记这些缺失的元组，用一个特殊的掩码元组 $g_{[m]} = ([m], [m], [m])$ 替换任何连续的缺失元组。得到的序列被称为输入稀疏轨迹，并被表示为 $\mathcal{T}^{(\text{in})} = \langle g_1^{\mathcal{T}'}, g_{[m]}, g_2^{\mathcal{T}'} \rangle$。

$\mathcal{T}^{(\text{in})}$ 中的每个元组对应于需要生成的 $\widetilde{\mathcal{T}}$ 的一部分。在图 7-1 中，$g_1^{\mathcal{T}'}$ 对应于 $\langle g_1^{\widetilde{\mathcal{T}}} \rangle$，$g_{[m]}$ 对应于 $\langle g_2^{\widetilde{\mathcal{T}}}, g_3^{\widetilde{\mathcal{T}}} \rangle$，$g_2^{\mathcal{T}'}$ 对应于 $\langle g_4^{\widetilde{\mathcal{T}}} \rangle$。将这些部分称为跨度。为了引导编码器按自回归方式生成每个跨度，遵循常规做法，构建一对源跨度和目标跨度。具体而言，每个跨度前面添加一个起始元组 $g_{[s]} = ([s], [s], [s])$ 作为输入源跨度，后面添加一个结束元组 $g_{[e]} = ([e], [e], [e])$ 作为输出目标跨度。从 $\widetilde{\mathcal{T}}$ 中构建的所有跨度构成源序列 $\mathcal{T}^{(\text{src})}$ 和目标稠密轨迹 $\mathcal{T}^{(\text{tgt})}$。由于自回归是单向的，为了保留学习模型提取双向相关性的能力，源跨度和目标跨度在拼接之前进行顺序的随机排列。

与 $\mathcal{T}^{(\text{in})}$ 和 $\mathcal{T}^{(\text{src})}$ 相关联的是双层位置索引。其中，第一层表示 $\mathcal{T}^{(\text{in})}$ 的元组位置和 $\mathcal{T}^{(\text{src})}$ 的源跨度的位置。第二层表示 $\mathcal{T}^{(\text{src})}$ 中每个源跨度内的元组位置。

最后，$\mathcal{T}^{(\text{in})}$ 作为已知信息输入模型，$\mathcal{T}^{(\text{src})}$ 作为自回归生成的指导，而 $\mathcal{T}^{(\text{tgt})}$ 则是生成的标签。两层位置索引也作为输入特征。这些序列的详细准备细节在第 7.2.2 节中提供。模型 f_θ 通过特征域编码层和层次掩码轨迹编码器实现，其设计在第 7.2.3 节和第 7.2.4 节中介绍。关于自监督训练和下游任务适配的全面解释在第 7.2.5 节中提供。

7.2.2　自监督学习样本的构建

(1) 稀疏轨迹重新采样

首先，从数据集中选择一条稠密轨迹 $\mathcal{T} \in \mathbb{T}$，其中其采样间隔 η 不超过 15 秒。为了提取其与道路段相关的信息，应用快速地图匹配（Fast Map Matching，FMM）

算法[131]，如下所示。

$$\widetilde{\mathcal{T}} = \text{FMM}(\mathcal{T}, \mathcal{G}) = \langle (\tilde{l}_1, t_1), (\tilde{l}_2, t_2), \ldots, (\tilde{l}_{|\mathcal{T}|}, t_{|\mathcal{T}|}) \rangle \tag{7-1}$$

同时，为了模拟稀疏轨迹的较长采样间隔，$\mathcal{T}$ 以间隔 μ 进行重新采样，其中 $\mu > \eta$，且 μ 是 η 的倍数，以得到 $\mathcal{T}$ 的对应稀疏轨迹。该过程可表示如下。

$$\mathcal{T}' = \langle (l_1, t_1), (l_{1+\mu/\eta}, t_{1+\mu/\eta}), (l_{1+2\mu/\eta}, t_{1+2\mu/\eta}), \ldots, (l_{|\mathcal{T}|}, t_{|\mathcal{T}|}) \rangle, \tag{7-2}$$

为了保持重新采样轨迹的完整性，$\mathcal{T}$ 中最后一个轨迹点被保留在 $\mathcal{T}'$ 中。

$\mathcal{T}'$ 中的每个轨迹点都存在于 $\widetilde{\mathcal{T}}$ 中。然而，与 $\widetilde{\mathcal{T}}$ 中的点 $(\tilde{l}_i, t_i)$ 相比，$\mathcal{T}'$ 中的点 (l_i, t_i) 缺少了路段域。使用一个特殊符号 $[m]$ 来表示该域的缺失，并将 $\mathcal{T}'$ 的第 i 个点按照定义 7.1 转换为一个元组：

$$g_i^{\mathcal{T}'} = ((l_j, \Omega(l_j)), t_j, [m]), j = 1 + (i-1)\mu/\eta, \tag{7-3}$$

其中，t_j 是归一化的一天中的时间，而 $\Omega(l_j)$ 表示与坐标 l_j 相关联的路段邻居集合，并被添加到该元组的空间域中。这增强了模型对空间坐标和路段之间关系的理解。形式化表示如下。

$$\Omega(l_i) = \{e_j, e_j \in \mathcal{E}, \text{dist}(l_i, e_j) \leqslant \delta\}, \tag{7-4}$$

其中 dist 是坐标点与路段中心之间的地球表面距离，δ 是一个以米为单位的距离阈值。

落在 $\mathcal{T}'$ 中连续点之间的 $\widetilde{\mathcal{T}}$ 中的子轨迹是不存在的。使用一个特殊的元组 $g_{[m]} = ([m], [m], [m])$ 来表示每个这样的子轨迹的缺失。这个元组被插入在普通元组 $g_i^{\mathcal{T}'}$ 和 $g_{i+1}^{\mathcal{T}'}$ 之间。

最后，输入稀疏轨迹 $\mathcal{T}^{(\text{in})}$ 被构建为一个元组序列：

$$\mathcal{T}^{(\text{in})} = \langle g_1^{\mathcal{T}'}, g_{[m]}, g_2^{\mathcal{T}'}, g_{[m]}, \ldots, g_{|\mathcal{T}'|}^{\mathcal{T}'} \rangle \tag{7-5}$$

(2) 源与目标对

每个普通元组 $g_i^{\mathcal{T}'}$ 对应着 $\widetilde{\mathcal{T}}$ 中的第 j 个轨迹点 $(\tilde{l}_j, t_j)$，其中 $j = 1 + (i-1)\mu/\eta$。点 $(\tilde{l}_j, t_j)$ 需要在自监督训练过程中生成。根据定义 7.1，可以将该点转化为一个包含单个元组的跨度：

$$\langle g_j^{\widetilde{\mathcal{T}}} \rangle = \langle (l_j, t_j, (e_j, r_j)) \rangle, \tag{7-6}$$

在图 7-1 中，元组 $g_1^{\mathcal{T}'}$ 对应于跨度 $\langle g_1^{\widetilde{\mathcal{T}}} \rangle$，而元组 $g_2^{\mathcal{T}'}$ 对应于跨度 $\langle g_4^{\widetilde{\mathcal{T}}} \rangle$。

另一方面，每个特殊的元组 $g_{[m]}$ 对应于 $\widetilde{\mathcal{T}}$ 的一个子轨迹，也需要被生成。可以仿照公式 (7-6) 将子轨迹中的每个点进行变换，并得到一个包含单个或多个元组的跨度。在图 7-1 中，元组 $g_{[m]}$ 对应于跨度 $\langle g_2^{\widetilde{\mathcal{T}}}, g_3^{\widetilde{\mathcal{T}}} \rangle$。

为了实现这些跨度的自回归生成，为每个要生成的跨度构建了一对源跨度和目标跨度。具体而言，给定一个跨度 $\langle g_j^{\widetilde{\mathcal{T}}} \rangle$，在其前面添加一个特殊的元组 $g_{[s]} = ([s],[s],[s])$，形成源跨度 $\langle g_{[s]}, g_j^{\widetilde{\mathcal{T}}} \rangle$，然后在其后面添加一个特殊的元组 $g_{[e]} = ([e],[e],[e])$，形成目标跨度 $\langle g_j^{\widetilde{\mathcal{T}}}, g_{[e]} \rangle$。这个过程也适用于包含多个元组的跨度。$g_{[s]}$ 表示一个跨度的开始，用 $g_{[e]}$ 表示一个跨度的结束。

构建了所有源跨度和目标跨度的配对后，它们的顺序经过随机排列，以便模型能够提取双向相关性。然后，通过排列后的源跨度和目标跨度，连接源序列 $\mathcal{T}^{(\text{src})}$ 和目标稠密轨迹 $\mathcal{T}^{(\text{tgt})}$。在图 7-1 中，$\mathcal{T}^{(\text{src})}$ 和 $\mathcal{T}^{(\text{tgt})}$ 由对应于待生成跨度 $\langle g_4^{\widetilde{\mathcal{T}}} \rangle$，$\langle g_2^{\widetilde{\mathcal{T}}}, g_3^{\widetilde{\mathcal{T}}} \rangle$ 和 $\langle g_1^{\widetilde{\mathcal{T}}} \rangle$ 的源跨度和目标跨度拼接。

(3) 双层位置索引

由于 $\mathcal{T}^{(\text{in})}$ 中的每个元组对应于 $\mathcal{T}^{(\text{src})}$ 中的一个跨度，遵循 GLM 的做法[130]，采用双层位置索引。

第一层位置表示 $\mathcal{T}^{(\text{in})}$ 中的元组与 $\mathcal{T}^{(\text{src})}$ 中的跨度之间的对应关系。具体而言，对于 $\mathcal{T}^{(\text{in})}$ 的第 i 个元组和 $\mathcal{T}^{(\text{src})}$ 的第 i 个跨度，第一层位置的计算如下所示。

$$\mathcal{P}_i^{1,(\text{in})} = i, \mathcal{P}_i^{1,(\text{src})} = \langle i, i, \ldots, i \rangle \tag{7-7}$$

第二层的位置表示每个 $\mathcal{T}^{(\text{src})}$ 跨度中元组的顺序。因此，对于 $\mathcal{T}^{(\text{in})}$ 的第 i 个元组和 $\mathcal{T}^{(\text{src})}$ 的第 i 个跨度，第二层的位置计算如下。

$$\mathcal{P}_i^{2,(\text{in})} = 0, \mathcal{P}_i^{2,(\text{src})} = \langle 1, 2, \ldots, N \rangle, \tag{7-8}$$

其中 N 是 $\mathcal{T}^{(\text{src})}$ 的第 i 个跨度的长度。

7.2.3 特征域编码

每个元组中的三个特征域包括连续数值特征和离散分类特征。本模型包含一个精心设计的特征域编码层，以增强模型从这些域中提取信息的能力。该层用于将特征域映射到潜在空间中。图 7-3 展示了该层的结构。

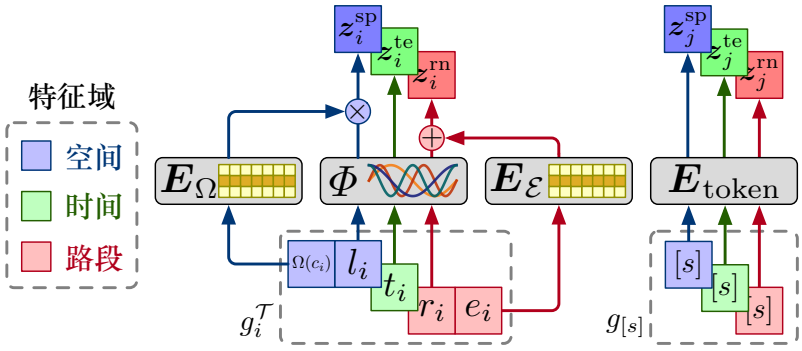


图 7-3　特征域编码层的示意图

Figure 7-3 Illustration of the feature domain encoding layer.

(1) 编码连续数值特征

连续特征通常表现出周期性特点。例如，个体的出行行为在一天中的相同时段或一周中的相同天通常表现出相似性。同时，两个轨迹点之间连续特征的相对距离可以揭示高层信息，例如两点之间的行驶距离和时间。

为了反映连续特征的这些特点，本模型的连续数值特征编码模块从可学习的傅里叶特征中汲取灵感[132,133]。对于给定的输入数值特征 $x \in \mathbb{R}$，编码模块 $\boldsymbol{\Phi}$ 将其投影到 d 维的潜在空间，定义如下：

$$\begin{aligned} \boldsymbol{\Phi} : \mathbb{R} \to \mathbb{R}^d, \boldsymbol{\Phi}(x) = \boldsymbol{W}_{\boldsymbol{\Phi}}[\cos(x\boldsymbol{v}_{\boldsymbol{\Phi}}) \| \sin(x\boldsymbol{v}_{\boldsymbol{\Phi}})] \\ x \in \{c_i^{\text{lng}}, c_i^{\text{lat}}, t_i, r_i\}, \end{aligned} \tag{7-9}$$

其中 $\boldsymbol{v}_{\boldsymbol{\Phi}} \in \mathbb{R}^{d/2}$ 表示可学习的投影向量，$\boldsymbol{W}_{\boldsymbol{\Phi}} \in \mathbb{R}^{d \times d}$ 表示可学习的投影矩阵。值得注意的是，对于四种类型的数值特征，分别使用不同的 $\boldsymbol{v}_{\boldsymbol{\Phi}}$ 和 $\boldsymbol{W}_{\boldsymbol{\Phi}}$ 参数。

通过 $\boldsymbol{\Phi}$，输入特征的周期性特征借助三角函数的属性得以保留。同时，考虑到 $\cos x_1 \cos x_2 + \sin x_1 \sin x_2 = \cos(x_1 - x_2)$，并且编码向量之间的距离通常由内积计算，$\boldsymbol{\Phi}$ 增强了模型识别特征相对距离的能力。这可以通过两条编码向量 $\boldsymbol{\Phi}(x_1)$ 和 $\boldsymbol{\Phi}(x_2)$ 的点积来证明：

$$\boldsymbol{\Phi}(x_1) \cdot \boldsymbol{\Phi}(x_2) = \|\boldsymbol{W}_{\boldsymbol{\Phi}}\|_2 \cos(x_1 - x_2) \|\boldsymbol{v}_{\boldsymbol{\Phi}}\|_1, \tag{7-10}$$

即两个值的编码向量之间的距离仅取决于可学习参数和这两个值之间的相对差异。

(2) 嵌入离散索引和特殊标记

为了将 $\mathcal{T}^{(\text{src})}$ 中的离散道路段索引投影到潜在空间，使用可学习矩阵 $\boldsymbol{E}_\mathcal{E} \in \mathbb{R}^{|\mathcal{E}| \times d}$ 定义的索引提取嵌入模块。道路段 s_i 的嵌入向量是为 $\boldsymbol{E}_\mathcal{E}$ 中的行向量，表示

为 $\boldsymbol{E}_{\mathcal{E}}(s_i)$。

同样，对于邻居集合 $\Omega(c_i)$ 中的路段索引，使用矩阵 $\boldsymbol{E}_{\Omega} \in \mathbb{R}^{|\mathcal{E}|\times d}$ 定义的另一个嵌入模块，用于计算集合中每个路段的嵌入向量。$\Omega(c_i)$ 中的路段嵌入集合表示为 $\Omega_E(c_i) = \{\boldsymbol{E}_{\Omega}(s_j), s_j \in \Omega(c_i)\}$。

对于特殊标记 $\{[m],[s],[e]\}$，在每个特征方面中为均每个标记指定一条嵌入向量，共计 9 条嵌入向量。这些向量组合在一起形成矩阵 $\boldsymbol{E}_{\text{token}} \in \mathbb{R}^{9\times d}$，表示为：

$$\boldsymbol{E}_{\text{token}} = \|_{i\in\{[m],[s],[e]\}}[\boldsymbol{e}_i^{\text{sp}}, \boldsymbol{e}_i^{\text{te}}, \boldsymbol{e}_i^{\text{rn}}], \tag{7-11}$$

其中 $\boldsymbol{e}_i^{\text{sp}}$、$\boldsymbol{e}_i^{\text{te}}$ 和 $\boldsymbol{e}_i^{\text{rn}}$ 分别是空间、时间和路网特征中特定标记的嵌入向量。

(3) 融合编码和嵌入向量

连续数值特征的编码向量和离散索引以及标记的嵌入向量经过融合，生成 $\mathcal{T}^{(\text{in})}$ 和 $\mathcal{T}^{(\text{src})}$ 中每个元组的一组潜在向量。

对于给定的元组 $((c_i,\Omega(c_i)),t_i,(e_i,r_i))$，空间、时间和路段域的潜在向量计算如下：

$$\begin{aligned}
\boldsymbol{z}'^{\text{sp}}_i &= \boldsymbol{\Phi}(c_i^{\text{lng}}) + \boldsymbol{\Phi}(c_i^{\text{lat}}) \\
\boldsymbol{z}_i^{\text{sp}} &= \boldsymbol{z}'^{\text{sp}}_i + \text{MultiHead}(\boldsymbol{z}'^{\text{sp}}_i, \Omega_E(c_i), \Omega_E(c_i)) \\
\boldsymbol{z}_i^{\text{te}} &= \boldsymbol{\Phi}(t_i) \\
\boldsymbol{z}_i^{\text{rn}} &= \boldsymbol{E}_{\mathcal{E}}(s_i) + \boldsymbol{\Phi}(r_i),
\end{aligned} \tag{7-12}$$

其中，MultiHead 表示 Transformer[73] 中定义的点积注意力机制，具有 N_h 个注意力头。中间潜在向量 $\boldsymbol{z}'^{\text{sp}}_i$ 被视为查询，邻居嵌入的集合 $\Omega_E(c_i)$ 被视为键和值。若任意特征域是特殊标记之一，该域的潜在向量计为相应特殊标记的嵌入向量。对于 $\mathcal{T}^{(\text{src})}$ 中的元组，由于不存在邻居集合 $\Omega(c_i)$，$\boldsymbol{z}'^{\text{sp}}_i$ 被视为最终的潜在向量 $\boldsymbol{z}_i^{\text{sp}}$。

最后，该元组的潜在向量元组表示为：

$$\boldsymbol{Z}_i = (\boldsymbol{z}_i^{\text{sp}}, \boldsymbol{z}_i^{\text{te}}, \boldsymbol{z}_i^{\text{rn}}) \tag{7-13}$$

7.2.4 层次掩码轨迹编码器

所提出模型中的分层掩码轨迹编码器被用于建模 $\mathcal{T}^{(\text{in})}$ 和 $\mathcal{T}^{(\text{src})}$ 中元组之间的相关性，并自回归地生成 $\mathcal{T}^{(\text{tgt})}$。其架构如图 7-4 所示。

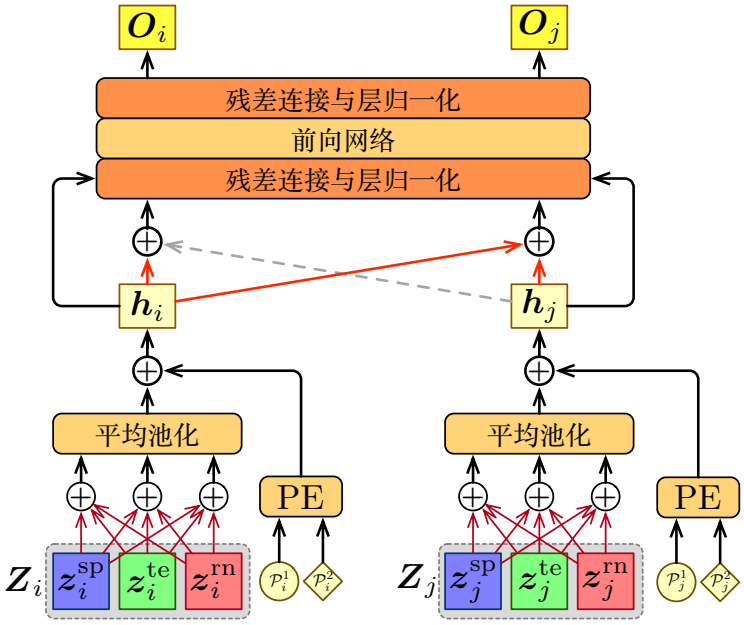


图 7-4　层次掩码轨迹编码器的结构

Figure 7-4 Architecture of the hierarchical masked trajectory encoder.

根据公式 (7-13)，$\mathcal{T}^{(\text{in})}$ 和 $\mathcal{T}^{(\text{src})}$ 中的每个元组都由一个潜在元组表示，该潜在元组包含表示三个特征域的潜在向量。为了理解元组中不同域之间的相互关系，使用以下的元组级自注意机制。

$$(\boldsymbol{h}_i^{\text{sp}}, \boldsymbol{h}_i^{\text{te}}, \boldsymbol{h}_i^{\text{rn}}) = \text{MultiHead}(\boldsymbol{Z}_i, \boldsymbol{Z}_i, \boldsymbol{Z}_i), \tag{7-14}$$

在计算过程中，$\boldsymbol{Z}_i$ 被视为一条长度为 3 的序列。随后，通过应用元组平均池化，结合公式 (7-7) 和 (7-8) 中定义的两层位置索引，计算该元组的隐藏状态，公式如下：

$$\boldsymbol{h}_i = \text{Mean}(\boldsymbol{h}_i^{\text{sp}}, \boldsymbol{h}_i^{\text{te}}, \boldsymbol{h}_i^{\text{rn}}) + \text{PE}(\mathcal{P}_i^1) + \text{PE}(\mathcal{P}_i^2), \tag{7-15}$$

其中 PE 表示 Transformer 中引入的位置编码[73]。

应用公式 (7-15) 到 $\mathcal{T}^{(\text{in})}$ 和 $\mathcal{T}^{(\text{src})}$ 中的每个元组后，得到对应的两条隐藏状态序列如下：

$$\begin{aligned} \boldsymbol{H}^{(\text{in})} &= \langle \boldsymbol{h}_1^{(\text{in})}, \boldsymbol{h}_2^{(\text{in})}, \ldots, \boldsymbol{h}_{|\mathcal{T}^{(\text{in})}|}^{(\text{in})} \rangle \\ \boldsymbol{H}^{(\text{src})} &= \langle \boldsymbol{h}_1^{(\text{src})}, \boldsymbol{h}_2^{(\text{src})}, \ldots, \boldsymbol{h}_{|\mathcal{T}^{(\text{src})}|}^{(\text{src})} \rangle \end{aligned} \tag{7-16}$$

为了从 $\boldsymbol{H}^{(\text{in})}$ 中提取信息，同时在 $\boldsymbol{H}^{(\text{src})}$ 的引导下实现 $\mathcal{T}^{(\text{tgt})}$ 的自回归生成，采用带有掩码的序列级自注意力机制。该注意力计算公式如下：

$$\begin{aligned} \boldsymbol{H}'^{(\text{in})} \| \boldsymbol{H}'^{(\text{src})} &= \text{MultiHead}(\boldsymbol{H}, \boldsymbol{H}, \boldsymbol{H}, \boldsymbol{M}) \\ \boldsymbol{H} &= \boldsymbol{H}^{(\text{in})} \| \boldsymbol{H}^{(\text{src})}, \end{aligned} \tag{7-17}$$

注意力掩码 $\boldsymbol{M} \in \mathbb{R}^{|\boldsymbol{H}|\times|\boldsymbol{H}|}$ 能够确保 $\boldsymbol{H}^{(\text{in})}$ 中的每个步骤都可以被 $\boldsymbol{H}$ 中的任何步骤关注，同时 $\boldsymbol{H}^{(\text{src})}$ 中的每个步骤只能被自身和其前驱步骤关注。具体来说，掩码的每个元素计算如下：

$$\boldsymbol{M}_{i,j} = \begin{cases} \text{True} & \text{如果 } j > |\boldsymbol{H}^{(\text{in})}| \text{ 且 } i < j \\ \text{False} & \text{其他情况} \end{cases} \tag{7-18}$$

遵循 Transformer 编码器的设计[73]，本模型的轨迹编码器的输出状态计算结合公式 (7-17) 获得的 $\boldsymbol{H}'^{(\text{src})}$，并引入了残差连接、层正则化和前馈网络。形式上表示为：

$$\begin{aligned} \boldsymbol{O}' &= \text{LayerNorm}(\boldsymbol{H}'^{(\text{src})} + \boldsymbol{H}^{(\text{src})}) \\ \boldsymbol{O} &= \text{LayerNorm}(\text{FFN}(\boldsymbol{O}') + \boldsymbol{O}'), \end{aligned} \tag{7-19}$$

其中 $\boldsymbol{O}$ 表示输出状态，FFN 是由两层全连接层组成的前馈网络。

最后，目标稠密轨迹 $\mathcal{T}^{(\text{tgt})}$ 的预测通过四个不同的预测模块完成，每个模块负责分别预测坐标 c_i、时间 t_i、道路段 s_i 和该路段上的行驶比例 r_i。形式上，对于 $\mathcal{T}^{(\text{tgt})}$ 中第 i 个元组的特征域的预测如下所示。

$$\begin{aligned} \hat{c}_i &= \boldsymbol{W}_c \boldsymbol{O}_i + \boldsymbol{b}_c \\ \hat{t}_i &= \boldsymbol{W}_t \boldsymbol{O}_i + \boldsymbol{b}_t \\ \hat{s}_i &= \text{argmax}(p(\hat{s}_i)),\ p(\hat{s}_i) = \text{Softmax}(\boldsymbol{W}_s \boldsymbol{O}_i + \boldsymbol{b}_s) \\ \hat{r}_i &= \boldsymbol{W}_r \boldsymbol{O}_i + \boldsymbol{b}_r, \end{aligned} \tag{7-20}$$

其中 $\boldsymbol{W}_c \in \mathbb{R}^{2\times d}$、$\boldsymbol{W}_t \in \mathbb{R}^{1\times d}$、$\boldsymbol{W}_s \in \mathbb{R}^{(|\mathcal{E}|+1)\times d}$ 和 $\boldsymbol{W}_r \in \mathbb{R}^{1\times d}$ 分别表示投影矩阵，而 $\boldsymbol{b}_c \in \mathbb{R}^2$、$\boldsymbol{b}_t \in \mathbb{R}^1$、$\boldsymbol{b}_s \in \mathbb{R}^{(|\mathcal{E}|+1)}$ 和 $\boldsymbol{b}_r \in \mathbb{R}^1$ 分别表示偏置。预测的道路段索引 $\hat{s}_i \in \mathcal{E} \cup \{[e]\}$ 同时充当对终止标志的预测。

7.2.5 自监督训练和下游任务适配

首先，本模型根据第 7.2.2 节中方案准备的样本进行自监督训练。随后，针对特定下游任务类型，使用特定的输入格式进行下游任务的适配。

(1) 自监督目标函数

对于给定的训练样本 $\{\mathcal{T}^{(\text{in})}, \mathcal{T}^{(\text{src})}, \mathcal{T}^{(\text{tgt})}\}$，根据预测的 $\widehat{\mathcal{T}}^{(\text{tgt})}$ 和预测目标 $\mathcal{T}^{(\text{tgt})}$，计算自监督目标函数。第 i 步的目标值计算如下：

$$\mathcal{L}_i = \mathbb{1}_i^{[e]} \left(\frac{1}{2}\|\hat{c}_i - c_i\|_2 + \|\hat{t}_i - t_i\|_2 + \|\hat{r}_i - r_i\|_2\right) - \log p(\hat{s}_i), \tag{7-21}$$

其中 $\mathbb{1}_i^{[e]}$ 指示第 i 步是否为终止标记，计算如下：

$$\mathbb{1}_i^{[e]} = \begin{cases} 0 & \text{如果 } \mathcal{T}_i^{(\text{tgt})} = [e] \\ 1 & \text{其他情况} \end{cases}, \tag{7-22}$$

然后，该样本的总自监督训练损失计算为目标序列上的损失的均值：

$$\mathcal{L}_{\mathcal{T}^{(\text{tgt})}} = \frac{1}{|\mathcal{T}^{(\text{tgt})}|} \sum_{i=1}^{|\mathcal{T}^{(\text{tgt})}|} \mathcal{L}_i \tag{7-23}$$

(2) 自监督训练

给定轨迹数据集 $\mathbb{T}$，通过选择采样间隔低于预定义阈值的轨迹来构建一个子训练集 $\mathbb{T}'$。在本章中，该阈值设置为 15 秒。对于每条轨迹 $\mathcal{T} \in \mathbb{T}'$，采用多种重采样间隔来生成多组训练样本。这个过程表示为：

$$\Lambda_{\mathcal{T}} = \bigcup_{\mu \in \Gamma} \{\mathcal{T}^{(\text{in})}, \mathcal{T}^{(\text{src})}, \mathcal{T}^{(\text{tgt})}\}, \tag{7-24}$$

其中，$\Lambda_{\mathcal{T}}$ 是从轨迹 $\mathcal{T}$ 派生的训练样本的集合，集合 Γ 表示所采用的不同重采样间隔。在本章中，$\Gamma = \{1\text{ 分钟}, 2\text{ 分钟}, 4\text{ 分钟}\}$。模型自监督训练阶段的优化目标表示为：

$$\underset{\theta}{\text{argmax}} \sum_{\mathcal{T} \in \mathbb{T}'} \sum_{\mathcal{T}^{(\text{tgt})} \in \Lambda_{\mathcal{T}}} -\mathcal{L}_{\mathcal{T}^{(\text{tgt})}}, \tag{7-25}$$

其中，θ 表示模型内所有可学习参数的集合。借助模拟的不同采样间隔的稀疏轨迹进行自监督训练，所提出的模型的通用性和泛化能力能得到提升。

(3) 下游任务适配

所提出的模型的灵活性使其可以在自监督训练后迁移到各种下游任务。本节详细介绍了模型迁移至第 2.4.3 节中介绍的三种不同类型下游任务所需的特定输入格式。

稀疏轨迹恢复： 此任务涉及到在第 7.2.2 节中定义的输入和输出序列。唯一的区别是 $\mathcal{T}^{(\text{src})}$ 和 $\mathcal{T}^{(\text{tgt})}$ 中的跨度不需要被随机排列。在稀疏轨迹中，连续点之间是否存在跨度可以通过采样间隔推断。具体而言，假设有轨迹点 (l_i, t_i) 和 (l_{i+1}, t_{i+1})，以及给定的采样间隔 η。如果 $t_{i+1} - t_i > \eta$，则在这两个点之间插入一个掩码元组 $g_{[m]}$，表示需要恢复一个缺失的跨度。

轨迹预测： 这个任务旨在根据历史轨迹点来预测未来的轨迹点。假设有历史轨迹 $\langle (l_1,t_1),(l_2,t_2),\ldots,(l_n,t_n)\rangle$，为模型提供相应的输入轨迹：

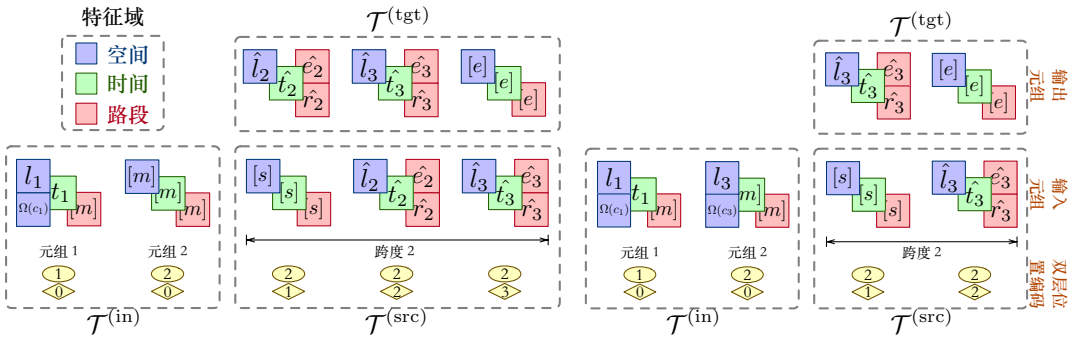
$$\mathcal{T}^{(\text{in})} = \langle g_1^{\mathcal{T}'},\ldots,g_n^{\mathcal{T}'},g_{[m]}\rangle = \langle (l_1,t_1,[m]),\ldots,(l_n,t_n,[m]),([m],[m],[m])\rangle \tag{7-26}$$

源序列以单个起始元组开始，即 $\mathcal{T}^{(\text{src})} = \langle ([s],[s],[s])\rangle$，模型以自回归的方式逐个元组地生成未来的预测，直到预测的道路段为结束符号，即 $\arg\max p(\hat{s}_i) = [e]$。图 7-5(a) 展示了该格式的一个示例。

起终点旅行时间估计： 这个任务涉及仅凭借起终点和出发时间来预测旅行时间。给定轨迹的起终点以及出发时间 — l_1，l_N 和 t_1，构建的输入轨迹如下：

$$\mathcal{T}^{(\text{in})} = \langle (c_1,t_1,[m]),(c_N,[m],[m])\rangle, \tag{7-27}$$

如图 7-5(b) 所示。由于预测的时间 $\hat{t}_i$ 被设计为相对于第一个轨迹点的时间偏移，因此 $\widehat{\mathcal{T}}^{(\text{tgt})}$ 中的最后一个轨迹点的预测时间就是估计的旅行时间。



(a) 轨迹预测　　(b) 旅行时间估计

图 7-5　多种下游任务的独特输入输出排布方案

Figure 7-5 Unique input and target arrangements for various downstream tasks.

7.3　实验

7.3.1　基线方法

为了验证所提出的模型在第 7.2.5 节中概述的三种下游任务中的有效性，其性能在实验中将与各种专门针对各自任务的最新方法进行比较。

稀疏轨迹恢复方法： 在稀疏轨迹恢复任务上，所提出的模型与与以下最新的稀疏轨迹恢复方法进行比较。

• **Shortest Path**[134]：在路网上通过 Dijkstra 最短路径算法[21] 恢复稀疏轨迹点之间的路径。

• **Linear**[135]**+HMM**[136]：使用线性插值来扩展稀疏轨迹，然后应用 HMM 算法。

• **TrImpute**[35]**+HMM**：基于群体智慧的稀疏轨迹插补方法。

• **DHTR**[137]**+HMM**：将 seq2seq 模型[138]、卡尔曼滤波器和 HMM 算法结合，恢复稠密轨迹。

• **AttnMove**[139]：利用注意力机制来预测道路段的序列。

• **MTrajRec**[37]：使用基于 GRU 的自回归模型恢复受道路网络约束的轨迹。

• **RNTrajRec**[36]：将 Transformer 模型与道路网络结构相结合，恢复轨迹。

轨迹预测方法： 在轨迹预测任务上，一些轨迹预测方法被作为比较的基线方法，包括端到端和自监督的方法。

• **trajectory2vec**[11]：构建行为序列以提取轨迹中的高层次关联性。

• **t2vec**[51]：在自编码模型的基础上构建，提供更强大的自监督轨迹学习模型。

• **DeepMove**[101]：一种端到端的基于注意力机制的循环神经网络模型，用于预测未来的移动轨迹。

• **Transformer**[73]：广泛使用的序列模型，擅长提取长序列中的关联性。

• **Trembr**[50]：将道路网络信息与自编码相结合，用于自监督轨迹学习。

• **START**[64]：结合了时空关联性和出行语义，用于自监督学习。

值得注意的是，*Trembr* 和 *START* 需要地图匹配的轨迹才能发挥最佳功能，而在稀疏轨迹的情境下直接获得这些轨迹是不可行的。因此，评估阶段使用了最有效的稀疏轨迹恢复基线方法 RNTrajRec 恢复的轨迹作为它们的输入。这些组合分别表示为 *Trembr+RNTR* 和 *START+RNTR*。同时，后续也将报告当它们被提供与地图匹配的真实稠密轨迹时的性能。

旅行时间估计方法： 在旅行时间估计任务中，仅有的输入信息是出发地、目的地和出发时间。因此，所提出的模型的性能与在类似条件下运行的一系列 OD 旅行时间估计方法进行比较。

• **RNE**[119]：通过参考它们的潜在嵌入来确定道路段之间的路径距离。

• **TEMP**[117]：计算在地点和时间上紧密对齐的历史轨迹的平均旅行时间。

• **LR**：基于训练标签，建立了从输入特征到旅行时间的线性映射。

- **GBM**：一种先进的非线性回归模型，通过 XGBoost[118] 实现。
- **ST-NN**[120]：同时预测给定 OD 对的旅行距离和旅行时间。
- **MURAT**[121]：利用出发时间作为额外信息，联合预测旅行距离和时间。
- **DeepOD**[122]：在训练期间考虑输入特征和历史轨迹之间的相关性。
- **DOT**：本文在第 5 章中提出的模型，生成轨迹的图像表示以估计旅行时间。

不使用自监督训练或微调的变体： 为了进一步评估所提出的模型中自监督训练和微调过程的有效性，引入两个额外的变体以供比较。

- **GTM w/o pt**：排除自监督训练过程，仅使用任务特定的数据来训练模型。
- **GTM w/o ft**：跳过微调过程，使模型能够在进行自监督训练后借助特定输入编排立即执行下游任务。

7.3.2 实验设置

GTM 及各基线方法在第 2.4.3 节中定义的轨迹预测、起终点旅行时间估计和稀疏轨迹恢复任务上验证。各任务的具体适配方案在第 7.2.5(iii) 节中详细介绍。对于稀疏轨迹恢复和轨迹预测任务，实验中模拟了三种稀疏轨迹的采样间隔，分别为 1 分钟、2 分钟和 4 分钟。稀疏轨迹恢复方法旨在将这些稀疏轨迹恢复为 15 秒采样间隔的稠密轨迹，而轨迹预测方法配置为根据稀疏轨迹除去最后一个点之外的历史，预测轨迹的目的地的道路段、路段上的前进比例、坐标和时间。

考虑到重采样稀疏轨迹需要采样相对稠密的原始轨迹数据的支持，实验中使用第 2.2.3 节中介绍的成都和波尔图出租车定位数据。对于两个数据集，其中的轨迹首先按照出发时间进行排序，然后按照 8:1:1 的比例划分为训练集、验证集和测试集。所有方法都使用训练集进行训练，使用验证集进行超参数调整和早停机制的实现。最终的衡量指标由测试集中重新采样的稀疏轨迹计算。

为了量化各种方法的有效性，针对不同的任务使用不同的度量标准。对于稀疏轨迹恢复任务，评估恢复的道路段的准确率（Precision(%)）和召回率（Recall(%)），以及坐标和道路地点的距离平均绝对误差（MAE），具体计算方法遵循 MTrajRec[37] 中的定义。对于轨迹预测任务，验证预测的路段索引的准确率（Precision(%)），坐标和道路地点的距离 MAE，以及时间的 MAE。对于旅行时间估计任务，使用均方根误差（RMSE）、MAE 和平均绝对百分比误差（MAPE）来评估估计的准确性。

所有方法均使用 Python 和 Pytorch[102] 实现。基准方法根据各自的论文建议的最佳参数设置进行配置。在实验过程中，所提出的模型的三个关键参数被考虑，它

们的范围和最佳值详见表 7-1。这些参数的影响在第 7.3.4 节中进一步研究。

表 7-1　超参数调整范围和最佳值
Table 7-1 Hyper-parameter range and optimal values.

参数	范围
d	32, 64, <u>128</u>, 192, 256
N_h	1, 2, 4, <u>8</u>, 16
δ (米)	10, 50, <u>100</u>, 150, 200

<u>下划线</u> 表示最佳参数设置。

7.3.3　总体性能对比

(1) 综合性能分析

表 7-2、 7-3 和 7-5 分别展示了在稀疏轨迹恢复、轨迹预测和旅行时间估计任务上各种方法之间的性能比较。结果清晰地表明，所提出的模型在各种任务中始终优于其他方法，显示了它在各种下游任务中的适应能力。

在稀疏轨迹恢复任务中，基线方法普遍的限制是它们被训练以适应具有特定采样间隔的稀疏轨迹，导致它们缺乏通用性。相反，通过自监督训练阶段，提出的模型能够适应具有不同采样间隔的轨迹，而无需完全重新训练。*GTM w/o ft* 的性能突显了在没有任何微调的情况下，该模型可以匹配甚至超越最佳的基线，展现了它面对不同采样间隔稀疏轨迹时的鲁棒性。

在轨迹预测任务中，尽管 *Trembr* 和 *START* 无疑是强大的轨迹学习方法，它们在稀疏轨迹中的效率显著下降。这是因为它们依赖于稠密的、地图匹配的轨迹以获得最佳功能。将它们与稀疏轨迹恢复方法配对并不会产生最佳结果，因为这种配置会导致误差累积。相反，提出的模型专门设计用于适应稀疏轨迹，并可以与下游任务无缝集成，从而优化了其准确性。表 7-4 显示，所提出的模型能够胜过使用真实稠密轨迹的 *Trembr* 和 *START*。

在旅行时间估计任务中，提出的模型超越了最先进的基线 *DOT*，这主要归功于其自监督训练增强了对起终点、轨迹和旅行时间之间相关性的理解。

表 7-2　各方法的稀疏轨迹恢复准确率比较

Table 7-2 Sparse Trajectory Recovery accuracy comparison of different approaches.

采样间隔 μ		1 分钟 / 2 分钟 / 4 分钟			
数据集	方法	Precision (%)	Recall (%)	MAE (坐标，米)	MAE (路网，米)
成都	Shortest Path	62.638 / 43.504 / 29.431	59.346 / 40.949 / 27.607	213.10 / 428.69 / 752.19	206.91 / 391.39 / 586.09
	Linear+HMM	66.642 / 48.604 / 36.209	65.557 / 45.234 / 30.496	183.64 / 385.23 / 675.85	169.46 / 378.96 / 564.19
	TrImpute+HMM	77.520 / 60.179 / 57.526	76.202 / 58.461 / 53.747	166.82 / 276.56 / 408.64	155.71 / 265.34 / 387.36
	DHTR+HMM	53.514 / 53.608 / 50.985	58.868 / 47.918 / 46.311	205.59 / 317.45 / 450.61	300.67 / 470.46 / 547.41
	AttnMove	84.162 / 81.402 / 78.645	81.839 / 76.612 / 69.257	252.59 / 280.20 / 354.39	201.51 / 258.69 / 323.52
	MTrajRec	85.039 / 82.596 / 80.684	83.351 / 80.113 / 72.952	243.01 / 264.15 / 311.53	173.67 / 204.58 / 282.88
	RNTrajRec	87.653 / 83.174 / 79.404	86.025 / 80.150 / 72.633	215.24 / 234.27 / 326.92	114.04 / 148.04 / 292.61
	GTM w/o pt	83.720 / 77.425 / 72.471	82.827 / 73.933 / 62.757	194.30 / 272.86 / 479.49	86.15 / 230.68 / 448.98
	GTM w/o ft	87.664 / 83.592 / 79.096	85.837 / 78.406 / 70.717	137.69 / 217.85 / 350.29	81.81 / 167.16 / 315.07
	GTM	**89.071 / 84.373 / 80.828**	**88.249 / 81.520 / 73.212**	**133.38 / 192.54 / 304.72**	**67.98 / 143.48 / 275.37**
波尔图	Shortest Path	69.780 / 53.590 / 40.492	60.354 / 46.263 / 33.758	202.17 / 434.37 / 679.72	165.02 / 319.32 / 478.66
	Linear+HMM	72.961 / 60.966 / 48.529	63.146 / 48.401 / 35.507	196.51 / 403.05 / 621.88	132.76 / 275.43 / 430.22
	TrImpute+HMM	76.781 / 66.492 / 50.021	69.599 / 58.676 / 43.052	132.76 / 275.43 / 430.22	128.48 / 235.63 / 347.30
	DHTR+HMM	63.287 / 58.897 / 52.658	62.511 / 56.444 / 42.462	235.32 / 292.65 / 355.23	285.68 / 336.48 / 389.18
	AttnMove	79.541 / 75.751 / 71.248	67.116 / 56.751 / 48.991	184.70 / 222.51 / 304.31	134.17 / 184.03 / 251.92
	MTrajRec	78.081 / 72.847 / 64.566	71.853 / 60.068 / 46.110	168.34 / 283.90 / 496.96	121.64 / 215.40 / 391.67
	RNTrajRec	80.305 / 77.094 / 75.573	74.953 / 65.370 / 50.965	135.17 / 175.42 / 294.08	111.75 / 152.49 / **230.30**
	GTM w/o pt	81.058 / 73.842 / 67.189	77.683 / 61.987 / 42.688	132.97 / 229.11 / 496.99	69.70 / 204.01 / 488.13
	GTM w/o ft	81.716 / 79.067 / 74.732	75.144 / 63.059 / 50.269	118.79 / 194.37 / 338.30	76.46 / 159.81 / 320.79
	GTM	**82.640 / 79.494 / 77.068**	**78.605 / 66.850 / 53.436**	**103.06 / 164.75 / 282.61**	**63.18 / 138.66** / 268.98

粗体表示最佳结果，下划线 表示次好结果。

表 7-3　各方法的轨迹预测准确率比较

Table 7-3 Trajectory Prediction accuracy comparison of different approaches.

采样间隔 μ		1 分钟 / 2 分钟 / 4 分钟			
数据集	方法	Accuracy (%)	MAE (坐标，米)	MAE (路网，米)	MAE (时间，秒)
成都	trajectory2vec	31.496 / 24.403 / 17.163	1514.5 / 1682.8 / 1861.6	1322.0 / 1616.5 / 1957.5	14.474 / 20.722 / 37.256
	t2vec	53.349 / 43.303 / 35.058	528.65 / 602.45 / 731.00	286.18 / 434.27 / 635.45	13.016 / 19.539 / 34.488
	DeepMove	58.499 / 45.985 / 37.338	319.14 / 461.73 / 664.93	258.96 / 397.95 / 607.42	11.994 / 19.435 / 35.039
	Transformer	68.192 / 66.028 / 65.139	374.36 / 402.11 / 431.01	236.86 / 287.61 / 300.92	16.287 / 29.848 / 34.226
	Trembr+RNTR	52.065 / 43.196 / 34.655	421.95 / 482.67 / 561.89	398.76 / 455.98 / 532.03	14.346 / 19.110 / 28.659
	START+RNTR	59.462 / 48.466 / 40.941	375.39 / 421.45 / 481.32	355.00 / 399.66 / 457.35	12.771 / 14.439 / 19.443
	GTM w/o pt	71.795 / 50.041 / 33.882	376.05 / 540.89 / 857.43	263.83 / 568.73 / 942.64	5.301 / 10.198 / 9.562
	GTM w/o ft	67.064 / 61.766 / 53.949	368.30 / 452.75 / 544.02	230.24 / 320.19 / 443.43	6.084 / 12.189 / 9.910
	GTM	**82.820 / 78.921 / 72.083**	**260.31 / 303.44 / 362.32**	**128.44 / 200.80 / 260.13**	**3.821 / 7.919 / 7.131**
波尔图	trajectory2vec	13.396 / 10.178 / 5.440	1709.3 / 2108.9 / 2488.5	2227.4 / 2425.6 / 3003.6	31.442 / 46.097 / 54.585
	t2vec	38.945 / 30.805 / 22.812	432.77 / 528.89 / 732.13	206.95 / 346.72 / 641.26	17.420 / 28.436 / 48.651
	DeepMove	43.774 / 33.562 / 23.645	252.22 / 390.57 / 679.62	197.89 / 328.93 / 681.40	16.998 / 26.309 / 46.629
	Transformer	43.441 / 41.425 / 40.685	323.58 / 351.18 / 402.32	216.18 / 236.89 / 253.72	18.231 / 30.541 / 49.083
	Trembr+RNTR	40.128 / 34.857 / 26.004	413.20 / 471.45 / 620.77	393.18 / 448.74 / 597.73	18.915 / 21.843 / 28.683
	START+RNTR	52.118 / 43.617 / 34.931	351.98 / 416.83 / 503.32	333.56 / 396.39 / 483.41	14.729 / 17.722 / 22.455
	GTM w/o pt	61.480 / 46.938 / 31.304	307.23 / 369.86 / 649.71	145.36 / 295.58 / 624.11	10.853 / 11.316 / 11.643
	GTM w/o ft	48.770 / 48.474 / 44.477	325.58 / 370.04 / 454.70	170.32 / 193.40 / 279.12	11.509 / 18.342 / 18.081
	GTM	**68.529 / 67.697 / 65.125**	**238.83 / 232.15 / 322.63**	**99.67 / 103.79 / 140.17**	**6.845 / 9.449 / 11.549**

粗体表示最佳结果，下划线 表示次好结果。

表 7-4　所提出的模型与基于稠密轨迹的基线模型之间的轨迹预测准确性比较

Table 7-4 Trajectory Prediction accuracy comparison between baselines fed with dense trajectories and the proposed model.

数据集		成都 / 波尔图		
方法	μ	Accuracy (%)	MAE (坐标，米)	MAE (时间，秒)
Trembr	15 秒	66.077 / 52.063	399.40 / 362.84	8.534 / 15.046
START	15 秒	74.990 / 62.997	354.76 / 318.26	8.068 / 12.476
GTM	**4 分钟**	72.083 / 65.125	362.32 / 322.63	7.131 / 11.549
GTM	**1 分钟**	**82.820 / 68.529**	**260.31 / 238.83**	**3.821 / 6.845**

粗体表示最佳结果，下划线 表示次好结果。

表 7-5　各方法的旅行时间估计准确率比较

Table 7-5 Travel Time Estimation accuracy comparison of different approaches.

数据集	成都 / 波尔图		
方法	MAE (分钟)	RMSE (分钟)	MAPE (%)
RNE	1.087 / 2.357	4.967 / 7.168	18.185 / 53.894
TEMP	0.816 / 2.610	1.100 / 3.414	13.003 / 59.178
LR	0.815 / 2.596	1.097 / 3.408	12.997 / 58.390
GBM	0.773 / 2.200	1.202 / 3.116	11.142 / 43.308
ST-NN	0.770 / 2.136	1.031 / 3.027	12.470 / 45.285
MURAT	0.731 / 1.971	0.979 / 2.827	11.931 / 41.259
DeepOD	0.640 / 1.899	0.880 / 2.780	10.517 / 36.956
DOT	0.614 / 1.777	0.841 / 2.644	9.937 / 34.883
GTM w/o pt	0.666 / 1.871	0.933 / 2.797	10.501 / 34.895
GTM w/o ft	2.203 / 3.470	2.469 / 4.694	35.039 / 45.700
GTM	**0.561 / 1.615**	**0.784 / 2.470**	**8.853 / 31.391**

粗体表示最佳结果，下划线 表示次好结果。

表 7-6　不同方法的效率指标

Table 7-6 Efficiency metrics of different approaches.

数据集		成都 / 波尔图		
任务	方法	模型大小 (MBytes)	训练时间 (分钟/轮)	测试时间 (秒)
STR	TrImpute+HMM	2.778 / 1.262	- / -	6.110K / 2.199K
	DHTR+HMM	6.426 / 6.426	0.176 / 0.115	2.503K / 0.257K
	AttnMove	6.799 / 6.250	6.844 / 2.460	92.673 / 32.081
	MTrajRec	19.180 / 18.495	8.470 / 4.437	0.125K / 54.062
	RNTrajRec	20.639 / 19.876	8.925 / 4.463	0.144K / 60.861
	GTM	10.146 / 9.333	1.668 / 1.470	21.984 / 45.237
TP	trajectory2vec	6.306 / 6.306	0.158 / 0.085	0.204 / 0.110
	t2vec	7.170 / 7.170	0.278 / 0.109	0.221 / 0.113
	DeepMove	5.282 / 5.282	0.253 / 0.126	0.567 / 0.292
	Transformer	6.295 / 6.295	1.090 / 0.538	3.568 / 1.596
	Trembr+RNTR	26.103 / 25.792	9.468 / 5.072	0.149K / 62.500
	START+RNTR	28.708 / 27.099	10.198 / 5.420	0.158K / 65.468
	GTM	10.146 / 9.333	1.667 / 1.450	3.357 / 1.703
TTE	RNE	2.446 / 2.173	0.100 / 0.040	0.170 / 0.062
	ST-NN	1.185 / 1.185	0.112 / 0.082	0.220 / 0.085
	MURAT	9.120 / 8.847	0.153 / 0.095	0.210 / 0.075
	DeepOD	8.184 / 7.928	0.382 / 0.171	0.328 / 0.099
	DOT	8.763 / 8.496	1.552 / 0.752	1.672 / 0.926
	GTM	10.146 / 9.333	0.533 / 0.336	1.231 / 0.647

(2) 效率比较

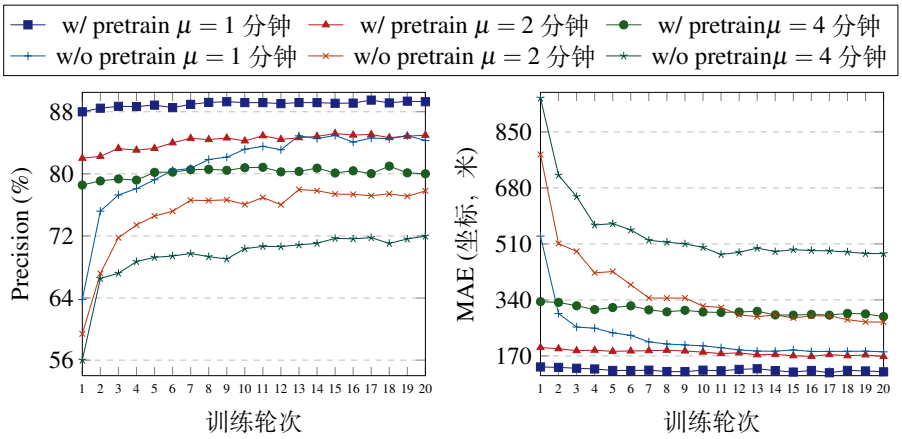
本节评估了不同方法在三个维度上的效率——模型大小、训练时间和测试时间。模型大小反映了这些方法在运行过程中的内存需求，而训练和测试时间则反映了关于这些方法在训练和测试阶段的效率。模型大小的计算基于每种方法内的可学习参数的类型和数量。此外，训练和评估速度是在配备有 Intel(R) Xeon(R) Gold 5215 CPU 和 nVidia(R) Quadro RTX 8000 GPU 的服务器上记录的。

正如表 7-6 中展示的，提出的模型在每个任务中与领先的方法相比均表现出

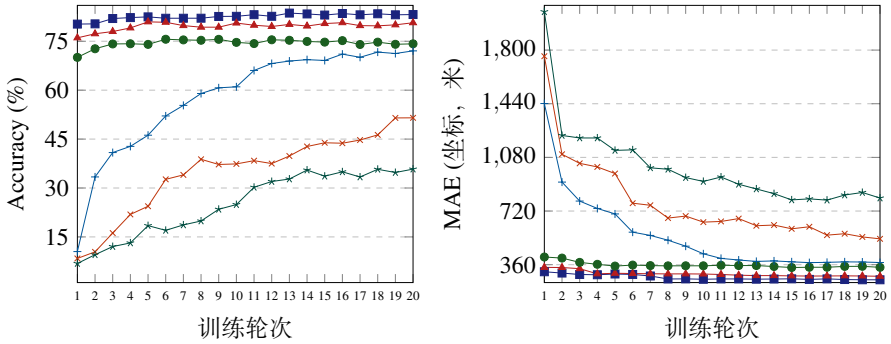
相近或更高的效率。值得注意的是，在实际应用场景中，提出的模型通常能展现更高的计算效率。这归因于它在各种下游任务中的通用性以及对不同采样间隔轨迹的适应性。例如，如果轨迹数据集包含具有 M 种不同采样间隔的稀疏轨迹，表中的所有稀疏轨迹恢复基准方法都需要进行 M 次训练以获得最佳性能，并需要存储 M 套模型参数。相比之下，所提出模型的自监督训练机制可以消除多次训练和冗余存储的需要。

7.3.4 模型分析

(1) 自监督训练的有效性



(a) 稀疏轨迹补全



(b) 轨迹预测

图 7-6 在成都数据集的测试集上验证的收敛速度

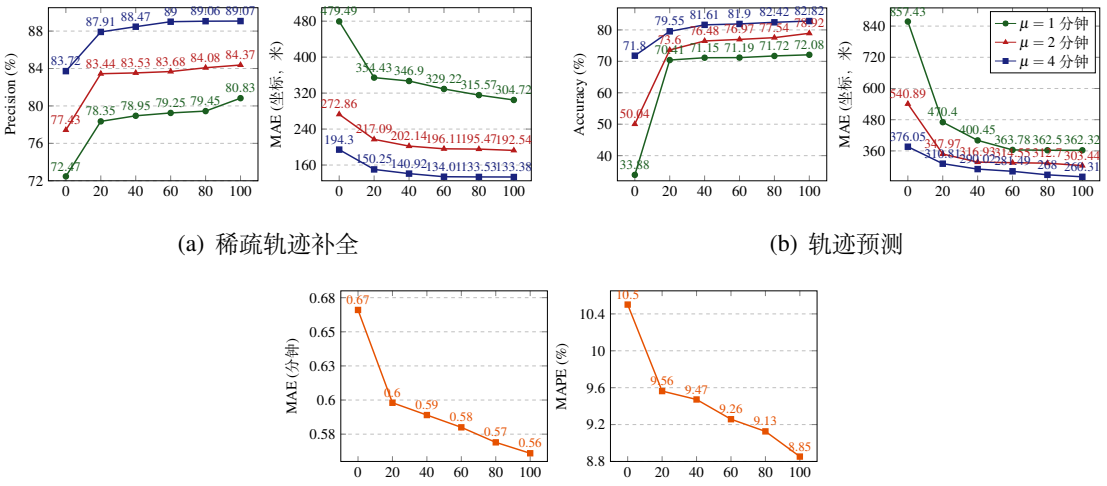
Figure 7-6 Convergence rate, measured on Chengdu testing set.

通过比较在使用和不使用自监督训练的情况下，所提出模型在下游任务上的收敛速度来评估自监督训练的有效性。具体来说，跟踪所提出模型在下游任务中

的性能指标在训练时期的变化，结果如图 7-6 所示。

观察结果明确：在所评估的下游任务中，模型的自监督训练版本始终在收敛速度上超过其未经自监督训练的对应版本。这突显了自监督学习模型的通用性，即其能够在多种下游任务之间实现无缝过渡，只需进行少量训练。通过在较少的训练周期内提供更高性能，自监督训练不仅提高了效率，还降低了计算负载，特别是在单一数据集用于多个任务的情况下。

(2) 有限数据下的自监督训练可扩展性



(a) 稀疏轨迹补全　　(b) 轨迹预测

(c) 旅行时间预测

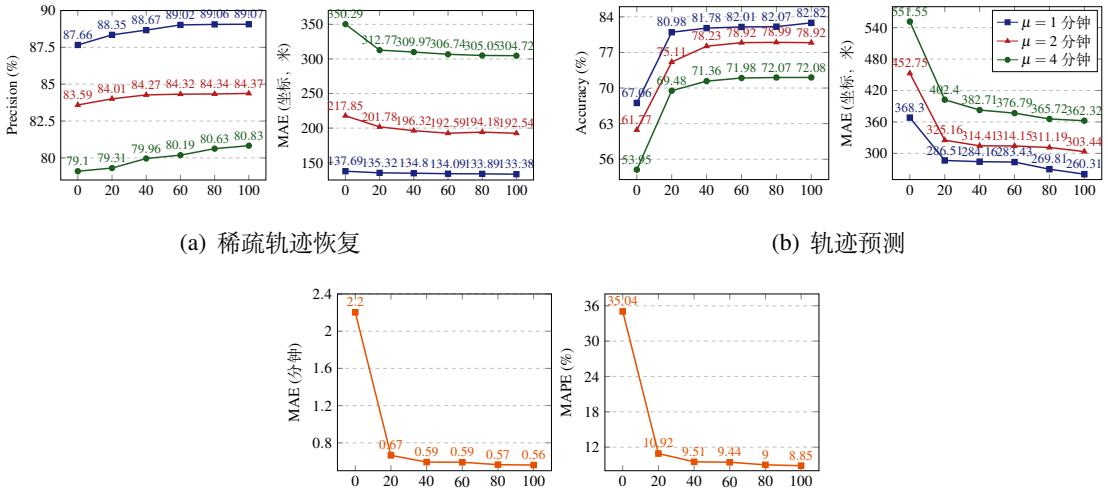
图 7-7　在成都数据集上验证的自监督训练数据集的可扩展性

Figure 7-7 Scalability with regard to pre-train sets (%), measured on Chengdu.

在所提出的模型中，构建自监督训练数据集需要一定数量的稠密轨迹。因此，一个直观的问题出现了：当面对有限数量的这些稠密轨迹时，该模型在多大程度上可以有效运行？为了评估这一方面，本节研究了在仅有训练集的一部分可用于自监督训练阶段的情况下，所提出模型的可扩展性。

图 7-7 展示了结果。值得注意的是，所提出模型表现出了很强的一致性。即使在自监督训练阶段仅使用有限数据，该模型的性能仍然保持强大。这突显了该模型在实际环境中的应用潜力与可行性，即使获取大规模稠密轨迹较为困难。此外，将 20% 规模训练集的结果与没有任何自监督训练（0% 规模）的模型相比，性能有了很大改善。这进一步证明了自监督训练过程的好处。

(3) 有限任务特定数据下的微调可扩展性



(a) 稀疏轨迹恢复　　(b) 轨迹预测

(c) 旅行时间预测

图 7-8　在成都数据集上验证的微调数据集的可扩展性

Figure 7-8 Scalability with regard to fine-tune sets (%), measured on Chengdu.

在表 7-3 和 7-5 中对所提出的模型在进行和不进行微调的性能进行的比较分析显示，前者具有明显优势，特别是在轨迹预测和旅行时间估计任务上。这可以归因于自监督训练和下游任务阶段之间的输入排布有较大的差异。这种显着差异引发了一个问题：该模型是否隐含地依赖于大规模的微调数据才能发挥最佳效果？为了解开这个谜团，本节探讨了在只有一部分标签下游数据可用于微调的情况下，该模型的可扩展性。

从图 7-8 中呈现的结果可以明显看出，即使只提供了完整微调数据的 20%，所提出的模型仍然能够在下游任务中接近其最佳性能。将这些结果与不进行微调的情况（0% 规模）进行对比，可以明显看出，即使小规模的微调数据也可以大幅提升该模型在下游任务上的准确性。这些发现展现了该模型在现实世界中的可操作性，考虑到收集大规模的任务特定标记数据集可能具有挑战性。

(4) 超参数的有效性

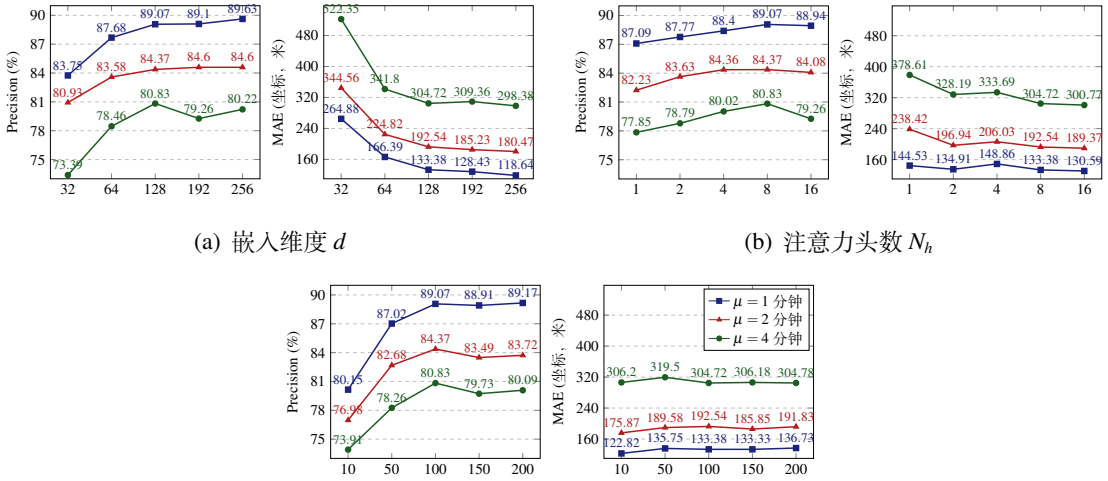
本节对表 7-1 中的三个主要超参数的影响进行了分析。在稀疏轨迹恢复任务的背景下，使用成都测试集来测量性能指标，特别关注 Precision（%）和 MAE（坐标，米）这两个指标。结果见图 7-9，有以下发现：

1）嵌入维度 d 决定了模型的表达能力。如图 7-9(a) 所示，d 的增加伴随着性

能提升。然而，在超过 $d = 128$ 的阈值后，性能提升有限，而计算和内存开销显著增加。因此，$d = 128$ 是性能和计算效率之间的理想折衷。

2）注意力头数 N_h 决定了模型中编码器的表征能力。从图 7-9(b) 中得出的结论是，对于精度和 MAE 指标，最有益的值为 $N_h = 8$。

3）距离阈值 δ 决定了道路段邻域的空间范围。从图 7-9(c) 中可以看到，其主要影响是在恢复的路段准确率上，对预测坐标的准确性几乎没有影响。在 $\delta = 100$ 处观察到路段准确率的峰值。使用较小的阈值可能会导致遗漏道路段候选项，而较大的阈值可能会引入更多的噪声。



(a) 嵌入维度 d　　(b) 注意力头数 N_h

(c) 距离阈值 δ (米)

图 7-9　在成都数据集的测试集上验证的超参数对稀疏轨迹恢复任务的影响

Figure 7-9 Effectiveness of hyper-parameters on the Sparse Trajectory Recovery task, measured on Chengdu test set.

(5) 消融研究

为了确定所提出的模型内各个特征和模块的影响，本节进行了消融研究。这涉及将完整模型的性能与其以下各个变种进行对比：

- **w/o neigh.**: 排除了 $\mathcal{T}^{(\text{in})}$ 中的道路段邻居集合 $\Omega(c_i)$。
- **w/o coor.**: 在 $\mathcal{T}^{(\text{in})}$ 中不使用坐标特征 c_i。
- **w/o time**: 在 $\mathcal{T}^{(\text{in})}$ 中省略时间特征 t_i。
- **w/o shuffle**: 在构建 $\mathcal{T}^{(\text{src})}$ 和 $\mathcal{T}^{(\text{tgt})}$ 时不对跨度进行随机排列。
- **Flat encoder**: 将 $\boldsymbol{h}_i$ 计算为 $\boldsymbol{Z}_i$ 的平均池化，而不是自注意力。

- **FC num. enc.**: 用全连接层替换数值编码模块 Φ。

与第 7.3.4 节中概述的方法一致，使用成都数据集的测试集，在稀疏轨迹恢复任务上计算结果。表 7-7 展示了这些结果，可以得出以下观察结论：

1）道路段邻居、坐标和时间特征共同增强了模型的有效性。省略其中任何一个特征都会导致性能明显下降。

2）在构建 $\mathcal{T}^{(\text{src})}$ 和 $\mathcal{T}^{(\text{tgt})}$ 时不进行随机排列操作会限制模型捕捉轨迹内双向相关性的能力，导致下游任务的精度下降。

3）将完整模型与其两个变种 *Flat encoder* 和 *FC num. enc.* 的结果进行比较，表明所提出模型内的层次掩码轨迹编码器和时空特征编码层提供了更优越的性能。

表 7-7　在成都数据集的测试集上验证的特征和模块在稀疏轨迹恢复任务上的有效性

Table 7-7 Effectiveness of features and modules on the Sparse Trajectory Recovery task, measured on Chengdu test set.

采样间隔 μ	1 分钟 / 2 分钟 / 4 分钟	
变体	Precision (%)	MAE (坐标，米)
w/o neigh.	76.834 / 74.494 / 71.950	128.36 / 193.41 / 300.30
w/o coor.	78.891 / 77.345 / 72.433	162.03 / 284.03 / 444.69
w/o time	86.821 / 83.230 / 77.055	140.98 / 203.23 / 318.81
w/o shuffle	84.406 / 77.467 / 73.589	158.95 / 276.26 / 433.86
Flat encoder	85.548 / 79.874 / 74.162	142.02 / 216.87 / 323.66
FC num. enc.	86.051 / 81.433 / 76.964	145.67 / 218.25 / 327.71
GTM	89.071 / 84.343 / 80.828	133.38 / 192.54 / 304.72

7.4 本章小结

本章为了强化自监督学习模型适配稀疏或不完整轨迹输入的能力，进行了面向通用性的轨迹自监督学习相关研究，并提出了一种新颖的通用轨迹模型 GTM。该模型通过自监督训练，能够灵活适配多样的轨迹输入，进而迁移至多种下游任务与场景并提升下游任务的准确率。

具体而言，GTM 首先将轨迹的时空特征分为时间、空间和路段三个域，其中每个域可独立掩盖与生成，因此 GTM 能够灵活适配不完整的轨迹输入。GTM 也包含特征域编码层和基于注意力的层次轨迹编码器，能够有效地从特征域中挖掘

时空特征。随后，GTM 通过从稀疏轨迹恢复对应稠密轨迹构建自监督学习目标，强化模型对稀疏轨迹的鲁棒性。GTM 凭借其通用性的设计，能够适配于多种下游任务和数据场景并提升性能。

GTM 在两个出租车定位数据集上进行验证，其自监督学习得到的模型被适配于轨迹预测、稀疏轨迹恢复、起终点旅行时间估计三种下游任务。与基线模型的对比证明 GTM 能够提升各种下游任务的准确率，验证了其自监督学习的有效性。同时，在有限的自监督训练和微调数据上的结果证明了 GTM 具有较好的可扩展性，对模块的组件分析验证了 GTM 中各模块的有效性。

8 总结与展望

8.1 论文工作总结

时空轨迹数据是智能交通系统和智慧城市的重要根基。时空轨迹数据的可用性日益提升，为轨迹数据挖掘和应用提供了充分的条件；而智能交通系统和智慧城市的建设，对轨迹数据挖掘的性能的需求也日益提升。为了构建有效的时空轨迹挖掘模型，提高轨迹数据在应用场景中的可用性，面向轨迹数据的自监督学习是一项基础而关键的任务。本文针对构建时空轨迹自监督学习方法的两方面挑战，即多方面时空信息的全面挖掘与融合、稀疏或不完整轨迹输入的灵活适配，提出了一系列创新的时空轨迹自监督学习模型，解决时空轨迹自监督学习方法所面临的挑战，实现能够应用于多种下游任务并提升任务准确率的时空轨迹挖掘模型。

本文的主要研究内容和贡献总结为如下几点。

1）围绕时空轨迹中的访问时间信息，提出了一种新颖的时间感知的地点嵌入模型 TALE。它能够将轨迹中的绝对访问时间信息融入自监督学习模型，更全面地建模地点的功能特征。TALE 在四个轨迹数据集和三种下游任务上进行验证。实验结果表明，与现有自监督学习方法相比，TALE 可以提高各种下游任务的性能。

2）围绕时空轨迹中的上下文信息，提出了一种新颖的上下文与时间感知的地点嵌入模型 CTLE。该方法考虑轨迹中的动态上下文信息和相对访问时间差信息，更精准地建模多功能地点的特定功能特征。CTLE 在两个轨迹数据集和轨迹预测任务上进行验证，结果证明了其自监督学习的有效性。

3）围绕时空轨迹中的行程信息，提出了一种新颖的起终点先验的轨迹行程生成模型 IGOP。IGOP 以起终点和出发时间为先验信息生成轨迹行程，从历史轨迹中建模轨迹行程与起终点的关联性。IGOP 在两个轨迹数据集和两种下游任务上进行验证，结果表明其准确率和可解释性优于现有方法。

4）针对多方面时空信息的全面挖掘与融合，提出了一种新颖的起终点与多视图最大熵嵌入模型 MMTEC。MMTEC 有效地挖掘并融合时空轨迹中的离散、连续时空两方面视图，并将轨迹映射为能提升多种下游任务性能的嵌入向量。MMTEC 在两个轨迹数据集和三个种游任务上进行验证，结果表明其比起现有方法能够输出更高质量的轨迹嵌入表示。

5）针对稀疏或不完整轨迹输入的灵活适配，提出一种新颖的通用轨迹模型 GTM。GTM 将轨迹特征划分为能够独立掩盖与生成的三个特征域，使得模型能够

灵活适配多样的轨迹输入。GTM 在两个轨迹数据集和三种下游任务上进行验证，结果证明了该模型的适应性、鲁棒性和可扩展性。

8.2 未来工作展望

考虑到时空轨迹数据的复杂性，以及对轨迹数据挖掘性能需求的日益提升，未来仍需要对本文所探讨的课题进行扩展。未来的研究方向可以着重于优化现有模型的结构和性能，同时拓展其在不同领域和应用场景中的适用性。针对已提出的自监督表示学习模型，可能的方向包括但不限于模型优化、应用领域拓展、数据集多样性增强、跨模态信息融合、可解释性和可视化分析、实时性和效率优化等。通过这些方面的研究，可以更好地推动时空轨迹数据自监督表示学习方法的发展，并为智能交通系统、智慧城市等领域的实际应用提供更有力的支持和解决方案。具体来说，未来的研究工作方向可以围绕如下几点开展。

1）**模型优化和性能提升**：在未来的研究中，可以进一步优化现有模型并探索新的模型架构，以提高自监督学习模型在时空轨迹数据上的表达能力和性能。可能的方法包括引入更多的时间维度信息、设计更复杂的结构以捕捉时空关系，或者使用不同的预训练策略来增强模型的泛化能力。

2）**应用领域扩展**：将自监督学习模型应用于更多领域和应用场景，如交通管理、城市规划、旅游推荐等。可以探索其在其他领域中的适用性，并针对特定领域的需求进行模型的定制和优化。

3）**数据集拓展和多样性增强**：扩展现有的数据集或创建新的数据集，以增加轨迹数据的多样性和规模。这可以帮助验证模型的泛化能力，并提高模型在各种场景下的适应性。

4）**跨模态信息融合**：探索如何有效地融合时空轨迹数据与其他传感器数据（如传感器数据、社交媒体数据等），以提升模型对多模态信息的学习能力，并拓展应用范围。

5）**可解释性和可视化分析**：进一步研究如何增强模型的可解释性，使其学习到的特征和信息更易于理解和解释。同时，开发可视化工具来帮助用户理解模型的决策过程和结果输出。

6）**实时性和效率优化**：针对实时数据处理需求，优化模型的计算效率和实时性能，使其能够在实时应用中进行高效处理和预测。

参考文献

[1] Kong D, Wu F. HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction [C]. In International Joint Conference on Artificial Intelligence, 2018: 2341–2347.

[2] 乔少杰, 韩楠, 朱新文, et al. 基于卡尔曼滤波的动态轨迹预测算法 [J]. 电子学报, 2018, 46 (2): 418.

[3] 乔少杰, 金琨, 韩楠, et al. 一种基于高斯混合模型的轨迹预测算法 [J]. 软件学报, 2015, 26 (5): 1048–1063.

[4] Li X, Cong G, Sun A, et al. Learning travel time distributions with deep generative model [C]. In World Wide Web Conference, 2019: 1017–1027.

[5] Hong H, Lin Y, Yang X, et al. Heteta: heterogeneous information network embedding for estimating time of arrival [C]. In SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020: 2444–2454.

[6] Jin G, Yan H, Li F, et al. Hierarchical neural architecture search for travel time estimation [C]. In International Conference on Advances in Geographic Information Systems, 2021: 91–94.

[7] 柴华骏, 李瑞敏, 郭敏. 基于车牌识别数据的城市道路旅行时间分布规律及估计方法研究 [J]. 交通运输系统工程与信息, 2012, 12 (6): 41–47.

[8] Liu Y, Zhao K, Cong G, et al. Online anomalous trajectory detection with deep generative sequence modeling [C]. In International Conference on Data Engineering, 2020: 949–960.

[9] Han X, Cheng R, Ma C, et al. DeepTEA: effective and efficient online time-dependent trajectory outlier detection [J]. VLDB Endowment, 2022, 15 (7): 1493–1505.

[10] 毛嘉莉, 金澈清, 章志刚, et al. 轨迹大数据异常检测: 研究进展及系统框架 [J]. 软件学报, 2016, 28 (1): 17–34.

[11] Yao D, Zhang C, Zhu Z, et al. Trajectory clustering via deep representation learning [C]. In International Joint Conference on Neural Networks, 2017: 3880–3887.

[12] 袁冠, 夏士雄, 张磊, et al. 基于结构相似度的轨迹聚类算法 [J]. 通信学报, 2011, 9.

[13] 龚玺, 裴韬, 孙嘉, et al. 时空轨迹聚类方法研究进展 [J]. 地理科学进展, 2011, 30 (5): 522–534.

[14] 栾丽华, 吉根林. 决策树分类技术研究 [J]. 计算机工程, 2004, 30 (9): 94–96.

[15] 杨学兵, 张俊. 决策树算法及其核心技术 [J]. 计算机技术与发展, 2007, 17 (1): 43–45.

[16] Myles A J, Feudale R N, Liu Y, et al. An introduction to decision tree modeling [J]. Journal of Chemometrics, 2004, 18 (6): 275–285.

[17] Song Y-Y, Ying L. Decision tree methods: Applications for classification and prediction [J]. Shanghai Archives of Psychiatry, 2015, 27 (2): 130.

[18] Keller J M, Gray M R, Givens J A. A fuzzy k-nearest neighbor algorithm [J]. IEEE Transactions

on Systems, Man, and Cybernetics, 1985 (4): 580–585.

[19] 张涛, 陈先, 谢美萍, et al. 基于 K 近邻非参数回归的短时交通流预测方法 [J]. 系统工程理论与实践, 2010 (2): 376–384.

[20] 于滨, 邬珊华, 王明华, et al. K 近邻短时交通流预测模型 [J]. 交通运输工程学报, 2012, 12 (2): 105–111.

[21] Johnson D B. A note on Dijkstra's shortest path algorithm [J]. Journal of the ACM, 1973, 20 (3): 385–388.

[22] Gallo G, Pallottino S. Shortest path algorithms [J]. Annals of Operations Research, 1988, 13 (1): 1–79.

[23] 陆锋. 最短路径算法: 分类体系与研究进展 [J]. 测绘学报, 2001, 30 (003): 269–275.

[24] Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35 (8): 1798–1828.

[25] Ji S, Pan S, Cambria E, et al. A survey on knowledge graphs: Representation, acquisition, and applications [J]. IEEE Transactions on Neural Networks and Learning Systems, 2021, 33 (2): 494–514.

[26] 刘知远, 孙茂松, 林衍凯, et al. 知识表示学习研究进展 [J]. 计算机研究与发展, 2016, 53 (2): 247–261.

[27] Saltzer J H, Reed D P, Clark D D. End-to-end arguments in system design [J]. ACM Transactions on Computer Systems, 1984, 2 (4): 277–288.

[28] Liu X, Zhang F, Hou Z, et al. Self-supervised learning: Generative or contrastive [J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 35 (1): 857–876.

[29] Devlin J, Chang M, Lee K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding [C]. In North American Chapter of the Association for Computational Linguistics, 2019: 4171–4186.

[30] Touvron H, Martin L, Stone K, et al. Llama 2: Open foundation and fine-tuned chat models [J]. CoRR, 2023, abs/2307.09288.

[31] Tian Y, Krishnan D, Isola P. Contrastive multiview coding [C]. In European Conference on Computer Vision, 2020: 776–794.

[32] Chen T, Kornblith S, Norouzi M, et al. A simple framework for contrastive learning of visual representations [C]. In International Conference on Machine Learning, 2020: 1597–1607.

[33] Li T, Chen L, Jensen C S, et al. TRACE: Real-time compression of streaming trajectories in road networks [J]. Proc. VLDB Endow., 2021, 14 (7): 1175–1187.

[34] Fang Z, He C, Chen L, et al. A lightweight framework for fast trajectory simplification [C]. In International Conference on Data Engineering, 2023: 2386–2399.

[35] Elshrif M M, Isufaj K, Mokbel M F. Network-less trajectory imputation [C]. In International Conference on Advances in Geographic Information Systems, 2022: 8:1–8:10.

[36] Chen Y, Zhang H, Sun W, et al. RNTrajRec: Road network enhanced trajectory recovery with

spatial-temporal transformer [J], 2023: 829–842.

[37] Ren H, Ruan S, Li Y, et al. MTrajRec: Map-constrained trajectory recovery via seq2seq multi-task learning [C]. In SIGKDD Conference on Knowledge Discovery and Data Mining, 2021: 1410–1419.

[38] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space [C]. In International Conference on Learning Representation, 2013.

[39] Zhou Y, Huang Y. Deepmove: Learning place representations through large scale movement data [C]. In International Conference on Big Data, 2018: 2403–2412.

[40] Yao Z, Fu Y, Liu B, et al. Representing urban functions through zone embedding with human mobility patterns [C]. In International Joint Conference on Artificial Intelligence, 2018: 3919–3925.

[41] Pauls A, Klein D. Faster and smaller n-gram language models [C]. In Annual Meeting of the Association for Computational Linguistics, 2011: 258–267.

[42] Zhao S, Zhao T, King I, et al. Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation [C]. In International Conference on World Wide Web Companion, 2017: 153–162.

[43] Feng S, Cong G, An B, et al. POI2Vec: Geographical latent representation for predicting future visitors [C]. In AAAI Conference on Artificial Intelligence, 2017: 102–108.

[44] Shimizu T, Yabe T, Tsubouchi K. Learning fine grained place embeddings with spatial hierarchy from human mobility trajectories [J]. CoRR, 2020, abs/2002.02058.

[45] Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations [J]. CoRR, 2018, abs/1802.05365.

[46] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial networks [J]. Communications of the ACM, 2020, 63 (11): 139–144.

[47] Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models [C]. In Advances in Neural Information Processing Systems, 2020: 6840–6851.

[48] Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks [J]. science, 2006, 313 (5786): 504–507.

[49] Preechakul K, Chatthee N, Wizadwongsa S, et al. Diffusion autoencoders: Toward a meaningful and decodable representation [C]. In Conference on Computer Vision and Pattern Recognition, 2022: 10609–10619.

[50] Fu T-Y, Lee W-C. Trembr: Exploring road networks for trajectory representation learning [J]. ACM Transactions on Intelligent Systems and Technology, 2020, 11 (1): 1–25.

[51] Li X, Zhao K, Cong G, et al. Deep representation learning for trajectory similarity computation [C]. In International Conference on Data Engineering, 2018: 617–628.

[52] Chen Z, Li K, Zhou S, et al. Towards robust trajectory similarity computation: Representation-based spatio-temporal similarity quantification [J]. World Wide Web, 2023, 26 (4): 1271–1294.

[53] Rombach R, Blattmann A, Lorenz D, et al. High-resolution image synthesis with latent diffusion models [C]. In Conference on Computer Vision and Pattern Recognition, 2022: 10674–10685.

[54] Liu H, Jin C, Yang B, et al. Finding top-k shortest paths with diversity [J]. IEEE Trans. on Know. and Data Eng., 2017, 30 (3): 488–502.

[55] Liu H, Jin C, Yang B, et al. Finding top-k optimal sequenced routes [C]. In International Conference on Data Engineering, 2018: 569–580.

[56] Pedersen S A, Yang B, Jensen C S. Anytime stochastic routing with hybrid learning [J]. Proc. VLDB Endow., 2020, 13 (9): 1555–1567.

[57] Pedersen S A, Yang B, Jensen C S. Fast stochastic routing under time-varying uncertainty [J]. VLDB J., 2020, 29 (4): 819–839.

[58] Guo C, Yang B, Hu J, et al. Context-aware, preference-based vehicle routing [J]. VLDB J., 2020, 29 (5): 1149–1170.

[59] Yang S B, Guo C, Yang B. Context-aware path ranking in road networks [J]. IEEE Trans. Knowl. Data Eng., 2022, 34 (7): 3153–3168.

[60] Li X, Cong G, Cheng Y. Spatial transition learning on road networks with deep probabilistic models [C]. In International Conference on Data Engineering, 2020: 349–360.

[61] van den Oord A, Li Y, Vinyals O. Representation learning with contrastive predictive coding [J]. CoRR, 2018, abs/1807.03748.

[62] Yan B, Zhao G, Song L, et al. PreCLN: Pretrained-based contrastive learning network for vehicle trajectory prediction [J]. World Wide Web, 2022: 1–23.

[63] Zhou F, Dai Y, Gao Q, et al. Self-supervised human mobility learning for next location prediction and trajectory classification [J]. Knowl. Based Syst., 2021, 228: 107214.

[64] Jiang J, Pan D, Ren H, et al. Self-supervised trajectory representation learning with temporal regularities and travel semantics [J]. CoRR, 2022, abs/2211.09510.

[65] Yang H, H Bell M G. Models and algorithms for road network design: a review and some new developments [J]. Transport Reviews, 1998, 18 (3): 257–278.

[66] 朱庆, 李渊. 道路网络模型研究综述 [J]. 武汉大学学报: 信息科学版, 2007, 32 (006): 471–476.

[67] 唐科萍, 许方恒, 沈才梁. 基于位置服务的研究综述 [J]. 计算机应用研究, 2012, 29 (12): 4432–4436.

[68] 刘经南, 方媛, 郭迟, et al. 位置大数据的分析处理研究进展 [J]. 武汉大学学报 (信息科学版), 2014, 39 (4): 379–385.

[69] Chao P, Xu Y, Hua W, et al. A survey on map-matching algorithms [C]. In ADC, 2020: 121–133.

[70] OpenStreetMap [EB/OL]. https://www.openstreetmap.org/.

[71] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions [C]. In Conference on Computer

Vision and Pattern Recognition, 2015: 1–9.

[72] Hochreiter S, Schmidhuber J. Long short-term memory [J]. Neural Computation, 1997, 9 (8): 1735–1780.

[73] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need [C]. In Advances in Neural Information Processing systems, 2017: 5998–6008.

[74] 吴禀雅, 魏苗. 从深度学习回顾自然语言处理词嵌入方法 [J]. 电脑知识与技术: 学术版, 2016 (12X): 184–185.

[75] Wang B, Wang A, Chen F, et al. Evaluating word embedding models: Methods and experimental results [J]. APSIPA Transactions on Signal and Information Processing, 2019, 8: e19.

[76] Rong X. word2vec Parameter Learning Explained [J]. CoRR, 2014, abs/1411.2738.

[77] Liu Q, Kusner M J, Blunsom P. A survey on contextual embeddings [J]. CoRR, 2020, abs/2003.07278.

[78] 陈佛计, 朱枫, 吴清潇, et al. 生成对抗网络及其在图像生成中的应用研究综述 [J]. 计算机学报, 2021, 44 (2): 347–369.

[79] Principle of maximum entropy [EB/OL]. https://en.wikipedia.org/wiki/Principle_of_maximum_entropy.

[80] Liu X, Wang Z, Li Y-L, et al. Self-supervised learning via maximum entropy coding [C]. In Advances in Neural Information Processing Systems, 2022.

[81] Altman N S. An introduction to kernel and nearest-neighbor nonparametric regression [J]. The American Statistician, 1992, 46 (3): 175–185.

[82] Cai X, Chen J, Xiao Y, et al. Fresh-product supply chain management with logistics outsourcing [J]. Omega, 2013, 41 (4): 752–765.

[83] Wang Y, Yin H, Chen T, et al. Passenger mobility prediction via representation learning for dynamic directed and weighted graphs [J]. ACM Trans. on Intelli. Sys. and Tech., 2021, 13 (1): 1–25.

[84] Litman T. Transportation cost and benefit analysis [J]. Victoria Transport Policy Institute, 2009, 31: 1–19.

[85] Crainic T G, Laporte G. Planning models for freight transportation [J]. European Journal of Operational Research, 1997, 97 (3): 409–438.

[86] Li Y, Deng D, Demiryurek U, et al. Towards fast and accurate solutions to vehicle routing in a large-scale and dynamic environment [C]. In Advances in Spatial and Temporal Databases, 2015: 119–136.

[87] Ruan S, Xiong Z, Long C, et al. Doing in one go: Delivery time inference based on couriers' trajectories [C]. In SIGKDD Conference on Knowledge Discovery and Data Mining, 2020: 2813–2821.

[88] Wen H, Lin Y, Wu F, et al. Package pick-up route prediction via modeling couriers' spatial-temporal behaviors [C]. In International Conference on Data Engineering, 2021: 2141–2146.

[89] Zhang J, Zheng Y, Sun J, et al. Flow prediction in spatio-temporal networks based on multitask deep learning [J]. IEEE Trans. on Know. and Data Eng., 2019, 32 (3): 468–478.

[90] Sang Y, Xie Z, Chen W, et al. TULRN: Trajectory user linking on road networks [J]. World Wide Web, 2022: 1–17.

[91] Liang Y, Ouyang K, Wang Y, et al. TrajFormer: Efficient trajectory classification with transformers [C]. In International Conference on Information & Knowledge Management, 2022: 1229–1237.

[92] Liang Y, Ouyang K, Yan H, et al. Modeling trajectories with neural ordinary differential equations [C]. In International Joint Conference on Artificial Intelligence, 2021: 1498–1504.

[93] Guo Q, Sun Z, Zhang J, et al. An attentional recurrent neural network for personalized next location recommendation [C]. In AAAI Conference on Artificial Intelligence, 2020: 83–90.

[94] Feng S, Li X, Zeng Y, et al. Personalized ranking metric embedding for next new POI recommendation [C]. In International Joint Conference on Artificial Intelligence, 2015: 2069–2075.

[95] Zhou C, Bai J, Song J, et al. ATRank: An attention-based user behavior modeling framework for recommendation [C]. In AAAI Conference on Artificial Intelligence, 2018: 4564–4571.

[96] Zhao P, Zhu H, Liu Y, et al. Where to go next: A spatio-temporal gated network for next POI recommendation [C]. In AAAI Conference on Artificial Intelligence, 2019: 5877–5884.

[97] Pan Z, Liang Y, Wang W, et al. Urban traffic prediction from spatio-temporal data using deep meta learning [C]. In SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019: 1720–1730.

[98] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality [C]. In Advances in Neural Information Processing Systems 26, 2013: 3111–3119.

[99] Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization [J]. Journal of Machine Learning Research, 2011, 12: 2121–2159.

[100] Chung J, Gülçehre Ç, Cho K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling [J]. CoRR, 2014, abs/1412.3555.

[101] Feng J, Li Y, Zhang C, et al. DeepMove: Predicting human mobility with attentional recurrent networks [C]. In World Wide Web Conference, 2018: 1459–1468.

[102] Paszke A, Gross S, Massa F, et al. PyTorch: An imperative style, high-performance deep learning library [C]. In Advances in Neural Information Processing Systems, 2019: 8024–8035.

[103] Maaten L v d, Hinton G. Visualizing data using t-SNE [J]. Journal of machine learning research, 2008, 9: 2579–2605.

[104] Dai Z, Yang Z, Yang Y, et al. Transformer-XL: Attentive language models beyond a fixed-length context [C]. In Conference of the Association for Computational Linguistics, 2019: 2978–2988.

[105] Xu D, Ruan C, Körpeoglu E, et al. Inductive representation learning on temporal graphs [C]. In International Conference on Learning Representations, 2020.

[106] Liu X, Liu Y, Li X. Exploring the context of locations for personalized location recommendations [C]. In International Joint Conference on Artificial Intelligence, 2016: 1188–1194.

[107] Wan H, Lin Y, Guo S, et al. Pre-training time-aware location embeddings from spatial-temporal trajectories [J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 34 (11): 5510–5523.

[108] Liu Q, Wu S, Wang L, et al. Predicting the next location: A recurrent model with spatial and temporal contexts [C]. In AAAI Conference on Artificial Intelligence, 2016: 194–200.

[109] Xiao S, Yan J, Yang X, et al. Modeling the intensity function of point process via recurrent neural networks [C]. In AAAI Conference on Artificial Intelligence, 2017: 1597–1603.

[110] Sankaranarayanan J, Samet H. Distance oracles for spatial networks [C]. In International Conference on Data Engineering, 2009: 652–663.

[111] Sankaranarayanan J, Samet H, Alborzi H. Path oracles for spatial networks [J]. Proc. VLDB Endow., 2009, 2 (1): 1210–1221.

[112] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional networks for biomedical image segmentation [C]. In Medical Image Computing and Computer-Assisted Intervention, 2015: 234–241.

[113] Liu Z, Mao H, Wu C-Y, et al. A convnet for the 2020s [C]. In Conference on Computer Vision and Pattern Recognition, 2022: 11976–11986.

[114] Hendrycks D, Gimpel K. Gaussian error linear units (gelus) [J]. CoRR, 2016, abs/1606.08415.

[115] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale [C]. In International Conference on Learning Representations, 2021.

[116] Gan Y, Zhang H, Wang M. Travel time estimation based on neural network with auxiliary loss [C]. In International Conference on Advances in Geographic Information Systems, 2021: 642–645.

[117] Wang H, Tang X, Kuo Y-H, et al. A simple baseline for travel time estimation using large-scale trip data [J]. ACM Trans. on Intelli. Sys. and Tech., 2019, 10 (2): 1–22.

[118] Chen T, Guestrin C. Xgboost: A scalable tree boosting system [C]. In SIGKDD Conference on Knowledge Discovery and Data Mining, 2016: 785–794.

[119] Huang S, Wang Y, Zhao T, et al. A learning-based method for computing shortest path distances on road networks [C]. In International Conference on Data Engineering, 2021: 360–371.

[120] Jindal I, Qin Z T, Chen X, et al. A unified neural network approach for estimating travel time and distance for a taxi trip [J]. CoRR, 2017, abs/1710.04350.

[121] Li Y, Fu K, Wang Z, et al. Multi-task representation learning for travel time estimation [C]. In SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018: 1695–1704.

[122] Yuan H, Li G, Bao Z, et al. Effective travel time estimation: When historical trajectories over

road networks matter [C]. In International Conference on Management of Data, 2020: 2135–2149.

[123] Tashiro Y, Song J, Song Y, et al. CSDI: Conditional score-based diffusion models for probabilistic time series imputation [C]. In Advances in Neural Information Processing systems, 2021: 24804–24816.

[124] Zhou H, Zhang S, Peng J, et al. Informer: Beyond efficient transformer for long sequence time-series forecasting [C]. In AAAI Conference on Artificial Intelligence, 2021: 11106–11115.

[125] Chen Y, Li X, Cong G, et al. Robust road network representation learning: When traffic patterns meet traveling semantics [C]. In International Conference on Information & Knowledge Management, 2021: 211–220.

[126] Linzen T, Dupoux E, Goldberg Y. Assessing the ability of LSTMs to learn syntax-sensitive dependencies [J]. Transactions of the Association for Computational Linguistics, 2016, 4: 521–535.

[127] Ba L J, Kiros J R, Hinton G E. Layer Normalization [J]. CoRR, 2016, abs/1607.06450.

[128] Kidger P, Morrill J, Foster J, et al. Neural controlled differential equations for irregular time series [J]. Advances in Neural Information Processing Systems, 2020, 33: 6696–6707.

[129] Lin Y, Wan H, Guo S, et al. Pre-training context and time aware location embeddings from spatial-temporal trajectories for user next location prediction [C]. In AAAI Conference on Artificial Intelligence, 2021: 4241–4248.

[130] Du Z, Qian Y, Liu X, et al. GLM: General language model pretraining with autoregressive blank infilling [C]. In Annual Meeting of the Association for Computational Linguistics, 2022: 320–335.

[131] Yang C, Gidofalvi G. Fast map matching, an algorithm integrating hidden Markov model with precomputation [J]. International Journal of Geographical Information Science, 2018, 32 (3): 547–570.

[132] Tancik M, Srinivasan P, Mildenhall B, et al. Fourier features let networks learn high frequency functions in low dimensional domains [J]. Advances in Neural Information Processing Systems, 2020, 33: 7537–7547.

[133] Li Y, Si S, Li G, et al. Learnable fourier features for multi-dimensional spatial positional encoding [J]. Advances in Neural Information Processing Systems, 2021, 34: 15816–15829.

[134] Chambers E W, Fasy B T, Wang Y, et al. Map-matching using shortest paths [J]. ACM Trans. Spatial Algorithms Syst., 2020, 6 (1): 6:1–6:17.

[135] Hoteit S, Secci S, Sobolevsky S, et al. Estimating human trajectories and hotspots through mobile phone data [J]. Comput. Networks, 2014, 64: 296–307.

[136] Newson P, Krumm J. Hidden Markov map matching through noise and sparseness [C]. In SIGSPATIAL International Symposium on Advances in Geographic Information Systems, 2009: 336–343.

[137] Wang J, Wu N, Lu X, et al. Deep trajectory recovery with fine-grained calibration using kalman

filter [J]. IEEE Transactions on Knowledge and Data Engineering, 2019, 33 (3): 921–934.

[138] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks [C]. In Advances in Neural Information Processing Systems, 2014: 3104–3112.

[139] Xia T, Qi Y, Feng J, et al. AttnMove: History enhanced trajectory recovery via attentional network [C]. In AAAI Conference on Artificial Intelligence, 2021: 4494–4502.