Towards Efficient and Comprehensive Urban Spatial-Temporal Prediction: A Unified Library and Performance Benchmark

Jiawei Jiang, Chengkai Han, Wenjun Jiang, Wayne Xin Zhao, *Member, IEEE*, Jingyuan Wang *Member, IEEE,*

Abstract—As deep learning technology advances and more urban spatial-temporal data accumulates, an increasing number of deep learning models are being proposed to solve urban spatial-temporal prediction problems. However, there are limitations in the existing field, including open-source data being in various formats and difficult to use, few papers making their code and data openly available, and open-source models often using different frameworks and platforms, making comparisons challenging. A standardized framework is urgently needed to implement and evaluate these methods. To address these issues, we provide a comprehensive review of urban spatial-temporal prediction and propose a unified storage format for spatial-temporal data called *atomic files*. We also propose LibCity, an open-source library that offers researchers a credible experimental tool and a convenient development framework. In this library, we have reproduced 65 spatial-temporal prediction models and collected 55 spatial-temporal datasets, allowing researchers to conduct comprehensive experiments conveniently. Using LibCity, we conducted a series of experiments to validate the effectiveness of different models and components, and we summarized promising future technology developments and research directions for spatial-temporal prediction. By enabling fair model comparisons, designing a unified data storage format, and simplifying the process of developing new models, LibCity is poised to make significant contributions to the spatial-temporal prediction field.

Index Terms—Spatial-Temporal Prediction, Open-source Library, Performance Benchmark

1 Introduction

In recent years, with the advancement of sensor technology in urban areas, a large amount of data can be collected, providing new perspectives for using artificial intelligence technologies to solve urban prediction problems [1]. Solving spatial-temporal prediction problems is crucial in urban computing, facilitating the management and decision-making processes of smart cities and improving residents' living standards. The urban spatial-temporal prediction has numerous applications, including congestion control [2], route planning [3], vehicle dispatching [4], and POI recommendation [5].

However, accessing and utilizing existing open-source spatial-temporal datasets can be challenging as they are stored in different formats, such as NPZ, PKL, H5, CSV, and others. Deep learning methods based on neural networks are the primary approach for spatial-temporal prediction due to their powerful feature extraction capabilities. As a result, numerous urban spatial-temporal prediction techniques have been proposed in the literature.

Unfortunately, we found that less than 30% of the papers published in 11 leading conferences and journals have made their code and data open source, which creates reproducibility challenges in the field [6]. Additionally, these models are often implemented under different platforms or frameworks, which makes it challenging to reproduce the results in a unified manner for researchers. In particular, the accuracy of prediction models on a specific dataset is sensitive to the choice of hyperparameters. Without a public and unified standard dataset for benchmarking model performance, it is increasingly difficult to measure the effectiveness of new spatial-temporal prediction methods and fairly compare the performance of different models [1], [7].

In contrast, domains like Computer Vision, Natural Language Processing, and Recommendation Systems have standardized datasets such as IMAGENET [8] and algorithm libraries like MMDetection [9] and RecBole [10]. Unfortunately, urban spatial-temporal prediction lacks these resources. Therefore, we urgently need to develop a standardized library that considers all aspects of urban spatial-temporal prediction.

To address these challenges, we provide a comprehensive review of urban spatial-temporal prediction and propose a unified storage format for spatial-temporal data, *i.e.*, atomic files. Based on this, we introduce LibCity [1], an open-source library that supports standardized measurement of models. Using LibCity, we conducted a series of experiments to validate the effectiveness of different models and components. By providing a standard library for urban spatial-temporal prediction, we aim to enhance reproducibility, comparability, and the advancement of this

- *Jingyuan Wang is with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China, and also with School of Economics and Management, Beihang University, Beijing 100191, China. E-mail: jywang@buaa.edu.cn*
- *Jiawei Jiang, Wenjun Jiang, and Chengkai Han are with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China. E-mail: {jwjiang, jwjun, ckhan}@buaa.edu.cn*
- *Wayne Xin Zhao is with the Gaoling School of Artificial Intelligence, Renmin University of China, Beijing 100872, China*

Corresponding Author: Jingyuan Wang.

1. https://github.com/LibCity

field.

The main features of LibCity can be summarized in four aspects:

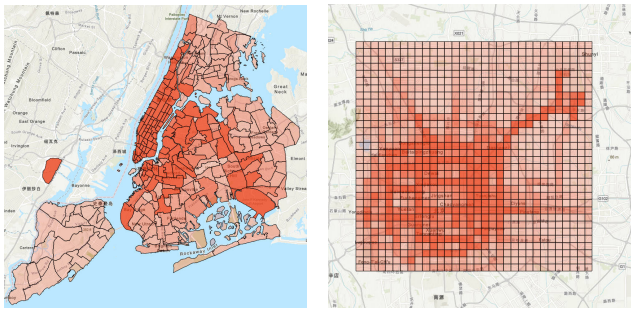
- **Unified and Modular Framework Design**: LibCity adopts a comprehensive and standardized approach to implementing, deploying, and evaluating spatial-temporal prediction models. The library is built entirely on PyTorch [11] and comprises five modules: Configuration, Data, Model, Evaluation, and Execution. Each module has a well-defined scope and collaborates seamlessly with others to deliver the complete functionality of the library. We design basic spatial-temporal data storage, unified model instantiation interfaces, and standardized evaluation procedure within these modules. Users can effortlessly train and evaluate existing models with simple configurations. On the other hand, developers can concentrate solely on the interfaces that are relevant to their models without worrying about the implementation details of other modules.
- **General and Extensible Data Storage Format**: Open-source spatial-temporal datasets are available in various storage formats. To provide a user-friendly interface and ensure the library's uniformity, LibCity has developed a general and extensible data storage format, namely *atomic files*, for urban spatial-temporal data. These atomic files consist of five categories, which represent the minimum information units in spatial-temporal data, and include Geographical Unit Data, User Unit Data, Unit Relation Data, Spatial-temporal Dynamic Data, and External Data. The atomic files are a generic and extensible structure that enables the representation of spatial-temporal data consistently. Using atomic files, LibCity has developed data processing functions and *Batch* extraction tools to create a unified data processing process, minimizing the effort required to develop new models. *Batch* is the standardized input format for models in LibCity.
- **Comprehensive Benchmark Tasks, Datasets and Models**: To increase the comprehensiveness of the dataset library, we have collected 55 widely used spatial-temporal datasets from 11 different countries covering various periods and processed them into the atomic files. We have also replicated 65 classic spatial-temporal prediction models, including state-of-the-art models, that cover three categories and nine sub-categories of tasks: *Macro Group Prediction Tasks* (*e.g.,* traffic flow prediction, traffic speed prediction, on-demand service prediction, traffic accident prediction, OD matrix prediction), *Micro Individual Prediction Tasks* (*e.g.,* trajectory next-location prediction, travel time prediction), and *Fundamental Tasks* (*e.g.,* map matching and road network representation learning). In addition, we have implemented rich auxiliary functions such as automatic parameter tuning and the visualization platform to facilitate the use of these datasets and models. We will continually incorporate more datasets and models into our library to provide more comprehensive benchmark tasks.
- **Diverse and Flexible Evaluation Metrics**: LibCity offers a range of standard evaluation metrics for assessing different types of spatial-temporal prediction models. These metrics cover typical tasks such as classification and regression, ensuring a comprehensive model performance evaluation. In addition, LibCity provides flexible evaluation strategies. For macro-level prediction tasks, data slicing and window settings are available to determine how the training, validation, and testing datasets are partitioned and the input/output data length for single-step and multi-step predictions. For micro-level prediction tasks, window settings enable the partitioning of trajectories to assess model performance on long, medium, and short trajectories. By combining the evaluation metric, dataset division, and window settings, users can conduct diverse and flexible evaluations of models belonging to the same task.

To the best of our knowledge, LibCity is the first open-source library for urban spatial-temporal prediction. We consider it an essential resource for exploring and developing spatial-temporal prediction models. By enabling fair model comparisons, designing a unified data storage format, and simplifying the process of developing new models, LibCity is poised to contribute to the spatial-temporal prediction field significantly. Additionally, LibCity helps to establish evaluation standards in the field and foster its fast-paced and standardized growth.

The main contributions of this paper are summarized as follows:

- We have categorized and summarized urban spatial-temporal data into basic units and designed atomic files as a unified storage format. To verify its effectiveness, we have successfully transformed 55 datasets into this format.
- We provide a comprehensive overview of urban spatial-temporal prediction, including macro-group prediction, micro-individual prediction, and fundamental tasks. For each task, we review the relevant technology and model development. We also categorize and summarize the 65 models replicated in this study based on their spatial-temporal modules.
- We propose LibCity, a unified, comprehensive, and extensible library for urban spatial-temporal prediction. We have also developed a web-based platform for experiment management and visualization to advance this field on the basis of LibCity.
- We conduct extensive experiments using LibCity, comparing the performance of nearly 20 models on nearly 20 datasets to create a comprehensive model performance leaderboard. Based on this, we identify promising future technology developments and research directions for spatial-temporal prediction.

The subsequent sections are organized as follows: Section 2 introduces the basic unit of spatial-temporal data and atomic file format, Section 3 introduces spatial-temporal data tasks and definitions, Section 4 introduces the development roadmap of models under different tasks, Section 5 introduces the LibCity open source library, Section 6 introduces the use cases of LibCity, Section 7 introduces the model comparison experiments under different tasks based on LibCity, Section 8 introduces the comparison of LibCity with existing open source libraries, and Section 9 provides a



(a) Irregular Regions in NYC (b) Regular Grids in Beijing

Fig. 1. Different division methods of plane based data.

conclusion.

2 Spatial-Temporal Data

2.1 Urban Spatial-Temporal Data

In modern cities, there are many urban information infrastructure devices such as Internet of Things (IoT) sensors, GPS terminals, smartphones, Location Based Services (LBS), Radio Frequency Identification (RFID), and wearable intelligent bracelets, which collect a lot of spatial-temporal big data related to the city.

The common urban spatial-temporal data contains three categories:

1) **Urban Scenes Data**: This kind of data is *low-frequency*, *static* urban structure data, such as urban map data, point-of-interest (POI) data, urban road network data, etc.
2) **Individual Behavior Data**: This kind of data is generally IoT data, which is *high-frequency* and *dynamic* spatial-temporal **trajectory** data, such as floating car GPS data, public transportation card data, cell phone signaling data, Location-Based Social Networks (LBSNs) data, etc.
3) **Group Dynamics Data**: This kind of data is generally the aggregated data of group behavior, which is *high-frequency*, *dynamic* spatial-temporal **attribute** data, such as population density, traffic road condition, ride demand, origin-destination (OD) network, climate, and weather data, etc.

Urban spatial-temporal data is characterized by its dynamic changes over time and space. The distribution of primary urban points of interest and road networks are examples of spatial-temporal static data. On the other hand, urban traffic flow data and weather data recorded by IoT sensors are considered spatial static and temporal dynamic data. In contrast, user trajectory data and check-in data are examples of spatial-temporal dynamic data, which capture the movements and behaviors of individuals over time and space.

2.2 Spatial-Temporal Data Unit

In this section, we define and classify urban spatial-temporal data in a normative way for the types of spatial-temporal data.

2.2.1 Basic Unit

According to the difference in spatial distribution, the basic geographical unit data are divided into three types:

- Point-based Data: This type of data mainly includes urban point of interest (POI) data, GPS sampling points, urban traffic sensors, traffic cameras, and other device data. We can use $(geo_i, \boldsymbol{l}_i, \boldsymbol{p}_i)$ to represent the point data, where geo_i is the ID of this point, $\boldsymbol{l}_i$ is the location of the point i, such as latitude and longitude information, and $\boldsymbol{p}_i$ is the attribute information of the point i, such as the POI category.
- Line-based Data: This type of data is mainly the road segment data. Similarly, we can use $(geo_i, \boldsymbol{l}_i, \boldsymbol{p}_i)$ to represent the line-based data, where geo_i is the ID of the road segment, $\boldsymbol{l}_i$ is the location of the road segment, which usually includes the latitude and longitude of the starting and ending locations, and $\boldsymbol{p}_i$ is the attribute information of road segment i, such as the average speed of the road segment.
- Plane-based Data: This type of data is mainly urban region data such as administrative districts. In particular, the city grid data obtained by the grid division of the city area is also plane-based data. The different methods of dividing the city area are given in Figure 1. Similarly, we can use $(geo_i, \boldsymbol{l}_i, \boldsymbol{p}_i)$ to represent the plane-based data, where geo_i is the ID of the region, $\boldsymbol{l}_i$ is the location of this region, which usually includes the latitude and longitude of the regional boundary locations, and $\boldsymbol{p}_i$ is the attribute information of region i, such as the traffic inflow or outflow of the region.

In addition to the macroscopic spatial distribution, a fundamental unit for spatial-temporal data (especially for trajectory data describing individual behaviors) is the user unit.

- User Data: A user unit is an unit that generates spatial-temporal behavior, and we can express a user's information by $(usr_i, \boldsymbol{p}_i)$, where usr_i is the ID of user i and $\boldsymbol{p}_i$ is the attribute information of user i, such as gender, age, etc.

2.2.2 Unit Relations

Unit relation data describes the relationships between units in urban spatial-temporal data, including three types:

- Relations between Geographical Units: They describe the physical relations between geographical units in urban space, *e.g.*, road connectivity and distance relationships between geographical units.
- Relations between User Units: They describe the social relationship between user units, *e.g.*, the friend relationship between user units.
- Relations between Geographical Units and User Units: They describe the attribution relationship between geographical units and user units, *e.g.*, the relationship between the user's residence and workplace.

In particular, since the relationships between geographic units have multiple structures, we further refine the relationships between geographic units as follows:

- Graph Relation Data: Common graph network data include urban sensor graph networks composed of traffic sensors, interest point graph networks composed

of POIs, urban road networks composed of road segments, and region graph networks composed of spatial regions/planes. In general, we can define such graph data as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{F}_{\mathcal{V}}, \boldsymbol{A})$, where $\mathcal{V}$ is a set of N nodes ($N = |\mathcal{V}|$), $\mathcal{N}_i$ is the neighborhood of the node v_i, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, $\boldsymbol{F}_{\mathcal{V}} \in \mathbb{R}^{N \times D}$ is a D-dimension attribute matrix, and $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is a weighted adjacency matrix between nodes.

- Grid Relation Data: A common Euclidean neighborhood data is the city grid data, which can be represented as $\boldsymbol{F}_{\boldsymbol{Grid}} \in \mathbb{R}^{I \times J \times D}$, where the city is divided into $I \times J$ grids, and each grid $Grid^{(i,j)}$ has D attributes. Each grid matches the distribution of the surrounding grids in the Euclidean neighborhood.
- OD Relation Data: This kind of data focuses on the relation between the origin and the destination, which is a more fine-grained relational data. We use $\boldsymbol{F}_{\mathcal{E}} \in \mathbb{R}^{N \times N \times D}$ to denote the D-dimension OD data between each pair of nodes in $\mathcal{V}$ (or say the D-dimension data of each edge), where each entry $\boldsymbol{F}_{\mathcal{E}}^{(i,j)} \in \mathbb{R}^D$ represents D-dimension attributes from node v_i to node v_j.
- Grid OD Relation Data: In particular, for the OD data between city grids, it can be denoted by $\boldsymbol{F}_{\mathcal{E}} \in \mathbb{R}^{I \times J \times I \times J \times D}$, and each entry $\boldsymbol{F}_{\mathcal{E}}^{(i_1,j_1),(i_2,j_2)}$ represents D-dimension attributes from $Grid^{(i_1,j_1)}$ to $Grid^{(i_2,j_2)}$.

Let us clarify the difference between Graph Relation Data and OD Relation Data. Graph Relation Data pertains to the attributes of the graph nodes, *i.e.,* $\boldsymbol{F}_{\mathcal{V}} \in \mathbb{R}^{N \times D}$, such as the traffic speed of each node, in a graph network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{F}_{\mathcal{V}}, \boldsymbol{F}_{\mathcal{E}}, \boldsymbol{A})$. On the other hand, OD Relation Data is concerned with the attributes of the graph edges, *i.e.,* $\boldsymbol{F}_{\mathcal{E}} \in \mathbb{R}^{N \times N \times D}$, which represents the transfer flow from the start of the edge to the end of the edge. This is typically used to model traffic flow between origin-destination pairs in transportation networks. Similarly, for Grid Relation Data, we can refine it into OD data between each pair of grids, where the attributes of the edges represent the traffic flow between the grids.

2.2.3 Unit Dynamics

Spatial-temporal dynamic data refers to the attribute information of geographical units in a city that changes dynamically over time. We can obtain complex spatial-temporal data by incorporating dynamic attributes in the temporal dimension into the basic geographical units and their relationships. There are two main categories of spatial-temporal data: Spatial Static Temporal Dynamic (SSTD) data and Spatial Dynamic Temporal Dynamic (SDTD) data, which will be introduced in detail below.

- Spatial Static Temporal Dynamic (SSTD) Data: In this type of data, the spatial attributes of a unit do not change over time, but other attributes of the unit change over time, such as the urban region traffic flow. Specifically, when the attributes of the Graph Network Relation Data change over time, we can obtain a spatial-temporal tensor, denoted as $\boldsymbol{X} \in \mathbb{R}^{T \times N \times D}$, which represents the D-dimension spatial-temporal attributes of N nodes at total T time slices. In particular, when the properties of the Grid Relation Data change over time, we can obtain a tensor $\boldsymbol{X} \in \mathbb{R}^{T \times I \times J \times D}$ which represents the D-dimension spatial-temporal attributes of the $I \times J$ grids at total T time slices. Similarly, for OD Relation Data, we use $\boldsymbol{X} \in \mathbb{R}^{T \times N \times N \times D}$ to denote the D-dimension spatial-temporal attributes of the $N \times N$ pairs of ODs at total T time slices. For the Grid OD Relation Data, we use $\boldsymbol{X} \in \mathbb{R}^{T \times I \times J \times I \times J \times D}$ to denote the D-dimension spatial-temporal attributes of the $I \times J \times I \times J$ pairs of grids at total T time slices.
- Spatial Dynamic Temporal Dynamic (SDTD) Data: In this type of data, a basic unit's spatial and temporal attributes change over time, such as the urban trajectory data. Specifically, a trajectory is a sequence of locations or sample points describing a user's underlying route over time. In other words, a trajectory can be seen as a time-ordered sequence composed of the positions of the basic geographical units (point, line, or plane units). Formally, we can define a trajectory as $\mathcal{T} = [\langle \boldsymbol{v}_i, t_i \rangle]_{i=1}^{m}$, where m is the length of the trajectory, each sample v_i in $\mathcal{T}$ can be viewed as any of the three basic geographical units described above and t_i is the visit timestamp for $\boldsymbol{v}_i$. For example, the basic unit of the *GPS-based Trajectory* is the sample points with latitude and longitude, the basic unit of the *POI-based Trajectory* is the POIs, the basic unit of the *Road-network Constrained Trajectory* is the road segments, and the basic unit of the *Region or Grid-based Trajectory* is the city regions or grids.

2.2.4 External Information

External data are environmental information describing the auxiliary information associated with the geographical and user units. For example, the city's weather and temperature information and calendar data (such as day of the week, time of day, and whether there are holidays) can be seen as external information. This information can aid in the accuracy of urban spatial-temporal predictions.

2.3 Atomic files

Existing open-source spatial-temporal datasets are usually stored in different formats, such as NPZ, PKL, H5, CSV, etc, which invariably increases the difficulty and burden for users to use these datasets. Therefore, to provide a unified representation and store format of urban spatial-temporal data, we define five types of *atomic files*, *i.e.,* five minimum information units of urban spatial-temporal data as follows:

- **Geographical Unit Data**: Geographical Unit data are the basic units in spatial-temporal data, *i.e.,* the point, the line, and the plane.
- **User Unit Data**: User Unit data describes attribute information of desensitized urban activity participants.
- **Unit Relation Data**: Unit Relation data describes the relationships between units in urban spatial-temporal prediction scenarios.
- **Spatial-temporal Dynamic Data**: Spatial-temporal dynamic data describes the attribute information of entities in a city that dynamically changes over time, including *Spatial Static Temporal Dynamic (SSTD) Data* and *Spatial Dynamic Temporal Dynamic (SDTD) Data*.
- **External Data**: External data describes the auxiliary information associated with urban geographical and user units.

For the above five types of atomic files, we use a comma-separated value format for data storage and define different file suffixes for different kinds of atomic files. In addition, we have restricted the information contained in each line in the atomic files. For example, the ".geo" file must contain ID, geographic entity type (point, line, polygon), and coordinate information. Other attributes must be stored after the above three columns, such as the POI category or the road width. More details can be found in Table 1.

3 Spatial-Temporal Prediction Tasks

The primary purpose of urban spatial-temporal prediction tasks is to make forecasts based on historical observations for urban spatial-temporal data, including *Spatial Static Temporal Dynamic (SSTD) Data* and *Spatial Dynamic Temporal Dynamic (SDTD) Data*. We classify the main spatial-temporal prediction tasks into two categories, one for **macro group prediction tasks** and one for **micro individual prediction tasks**. In addition to these two types of spatial-temporal prediction tasks in this study, some **fundamental tasks** that support urban spatial-temporal prediction are also considered. Specifically, the tasks targeted in this study are as follows:

3.1 Macro Group Prediction Tasks

This type of task is mainly for the *Spatial Static Temporal Dynamic (SSTD) Data*. Specially, these tasks are used to predict the macro group's spatial-temporal attributes in the future. Formally, given the *SSTD* data, our goal is to learn a mapping function f from the previous T steps' observation value to predict future T' steps' attributes [7], [12],

$$[\boldsymbol{X}_{(t-T+1)}, \cdots, \boldsymbol{X}_t; \mathcal{G}] \xrightarrow{f} [\boldsymbol{X}_{(t+1)}, \cdots, \boldsymbol{X}_{(t+T')}]. \quad (1)$$

where the tensor $\boldsymbol{X}$ can be obtained from Graph Relation Data, Grid Relation Data, OD Relation data, and Grid OD Relation Data.

Typical macro group prediction tasks include traffic flow prediction, traffic speed prediction, on-demand service prediction, origin-destination matrix prediction, and traffic accidents prediction as follows:

- *Traffic Flow Prediction* [13], [14] aims to forecast the number of vehicles that will enter or exit specific regions or road segments in a future time window. Accurate predictions can significantly improve dynamic traffic control, route planning, navigation services, and other high-level applications.
- *Traffic Speed Prediction* [15], [16] aims to forecast the average speed of vehicles over a specific road segment in the future. Here traffic speed refers to the distance traveled per unit of time, and the focus is on the average speed of vehicles across a particular road segment rather than on specific vehicles. This task is similar to traffic flow prediction, and both problems can be solved using similar prediction strategies.
- *On-demand Service Prediction* [17], [18] aims to forecast the number of ride requests for a particular region or road segment. Short-term demand prediction is critical for on-demand ride-hailing platforms like Uber and Didi because dynamic pricing depends on real-time demand prediction, and dispatch systems can relocate drivers to high-demand areas. Typically, the number of pick-ups and drop-offs represents the demand in a specific region during a particular time interval.
- *Origin-Destination Matrix Prediction* [19], [20] aims to forecast transitions between different nodes, also known as edge flow or origin-destination-based flow. OD prediction provides more detailed insight into travel needs and enhances understanding of urban traffic patterns compared to other traffic prediction tasks.
- *Traffic Accident Prediction* [21], [22] is crucial for improving public safety. However, predicting the occurrence of traffic accidents in terms of their location and time is generally difficult. In recent studies, researchers have shifted their focus to forecasting the number of traffic accidents or the severity of traffic risks in specific regions. Such regions with the highest risk level can be considered hot spots, allowing for targeted safety measures to be implemented.

3.2 Micro Individual Prediction Tasks

This type of task is mainly for the *Spatial Dynamic Temporal Dynamic (SDTD) Data*. Specially, these tasks perform task-specific label prediction based on the user's historical trajectory data. Formally, given the *SDTD* data, our goal is to learn a mapping function f from user's historical trajectory $\mathcal{T}$ to predict a task-specific label [23], [24],

$$\mathcal{T} = [\langle \boldsymbol{v}_i, t_i \rangle]_{i=1}^{m} \xrightarrow{f} label. \quad (2)$$

where the sample point $\boldsymbol{v}_i$ can be the Point, Line, or Plane Geographical Unit mentioned above. In other words, the trajectory $\mathcal{T}$ can be the GPS-based trajectories, POI-based trajectories, Road-network Constrained trajectories, and Region/Grid-based trajectories.

Typical micro individual prediction tasks include trajectory next-location prediction and travel time prediction (also called Estimated time of arrival, ETA) as follows:

- *Trajectory Next-Location Prediction* [25], [26] aims to predict the location that a user may visit next, given the historical trajectory of this user. Formally, given the historical trajectory $\mathcal{T} = [\langle \boldsymbol{v}_i, t_i \rangle]_{i=1}^{m}$ of user i, our goal is to estimate the probability of location $\boldsymbol{v}_{m+1}$ for user i in the next timestamp t_{m+1}. Most of the next-location prediction research focuses on the POI-based trajectories.
- *Travel Time Prediction* [27], [28] aims to predict the arrival time given the trajectory and departure time. Given the trajectory sequence $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$ and the departure time t_1, our goal is to estimate the travel time between the starting point $\boldsymbol{v}_1$ and the destination $\boldsymbol{v}_m$ through $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$. Besides, in some ETA scenarios, only the starting point, departure time, and destination are given without the specific trajectory sequence. Most of the travel time prediction research focuses on the GPS-based trajectories, Road-network Constrained trajectories, and Region or Grid-based trajectories.

3.3 Fundamental tasks

Fundamental tasks provide support for macro and micro prediction tasks mentioned above. The fundamental tasks

TABLE 1
Summary of Atomic Files

Suffix	Content
.geo	Geographical Unit Data
.usr	User Unit Data
.rel	Unit Relation Data
.dyna	Spatial Dynamic Temporal Dynamic (SDTD) Data for User Unit (Trajectories)
.dyna	Spatial Static Temporal Dynamic (SSTD) Data of Graph Network Unit Relation Data
.grid	Spatial Dynamic Temporal Dynamic (SDTD) Data for Grid Relation Data
.od	Spatial Dynamic Temporal Dynamic (SDTD) Data for OD Relation Data
.gridod	Spatial Dynamic Temporal Dynamic (SDTD) Data for Grid OD Relation Data
.ext	External Data

considered in this work include map matching and road network representation learning.

- *Map Matching* [29], [30] aims to match the raw GPS trajectories to the road segments of the road network. Given a GPS-based trajectory $\mathcal{T}^{gps}$ and a road network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{F}_{\mathcal{V}}, \boldsymbol{A})$, our goal is to find a road-network constrained trajectory $\mathcal{T}$ that matches $\mathcal{T}^{gps}$ with its real path.
- *Road Network Representation Learning* [31], [32] aims to discover the representations from raw road network data. Given a road network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{F}_{\mathcal{V}}, \boldsymbol{A})$, where $\boldsymbol{F}_{\mathcal{V}} \in \mathbb{R}^{N \times D}$ contains raw attributes, our goal is to derive a d-dimensional representation $\boldsymbol{F}_{\mathcal{V}}^{dense} \in \mathbb{R}^{N \times d}$ for each road segment on $\mathcal{G}$, where $d \ll D$. $\boldsymbol{F}_{\mathcal{V}}^{dense}$ is supposed to preserve the multifaceted characteristics of $\boldsymbol{F}_{\mathcal{V}}$.

4 Spatial-Temporal Prediction Models

4.1 Macro Group Prediction Tasks

In this section we give a review of the technical development roadmap of urban spatial-temporal prediction models. The urban spatial-temporal prediction task focuses on capturing spatial-temporal dependencies in data to achieve accurate forecasts. Traditional statistical methods or machine learning methods, such VAR [33], ARIMA [34] and SVR [35] are based on linear time-series analysis techniques and cannot effectively capture spatial-temporal dependencies in data. Recently, the emergence of deep learning has allowed researchers to explore its potential in urban spatial-temporal prediction. Deep learning methods can fully complement the spatial-temporal dependencies in the data. The following section provides a detailed summary of the spatial dependency modeling, temporal dependency modeling, and spatial-temporal fusion techniques to lay the foundation for the experiments of the technical exploration.

4.1.1 Spatial Dependencies

Spatial dependencies are present in urban spatial-temporal data, as per the first law of geography, that "everything is related to everything else, but things that are close to each other are more closely related." Moreover, due to the division of urban structural and functional areas, urban spatial-temporal data has a spatial hierarchy. Therefore, even distant locations may exhibit strong spatial dependencies due to the similarity of functions.

- Convolutional Neural Networks (CNNs) are widely used for modeling Grid Relation Data as these data are Euclidean data. Researchers often divide cities into grids based on latitude and longitude, which allows the spatial-temporal attribute data to be viewed as images. In this way, CNNs can extract spatial dependencies from different grids [17], [36], [37], [38]. For instance, ST-ResNet [14] utilizes a deep residual CNN network for traffic flow prediction.
- Graph Neural Networks (GNNs): GNNs are widely used for modeling Graph Relation Data due to their strong ability to represent graphs [15], [16], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54]. Among GNNs, graph convolutional neural networks (GCNs) [55] are the most commonly used for spatial-temporal prediction tasks. An essential issue in graph convolution models is the construction of the graph adjacency matrix, and early studies typically used static adjacency matrices [15], [16]. However, predefined graph adjacency matrices may not accurately reflect the actual spatial dependencies in spatial-temporal data. To address this, adaptive graph generation modules have been proposed [46], [56]. Additionally, some studies have focused on learning dynamic spatial dependencies to model the effects of changes in the data over time [43], [50], [57]. To model continuous spatial dependencies, Graph ODE-based models have been used for spatial-temporal prediction [58].
- Spatial Attention Mechanisms: The spatial attention mechanism plays an essential role in modeling time-varying spatial dependencies between geographical entities [37], [57], [59], [60], [61], [62], [63], [64], [65], [66]. Through the spatial attention, models can autonomously learn the dynamic spatial dependencies embedded in the spatial-temporal data by assigning adaptive weights to different locations at different time steps. For instance, GMAN [60] proposes both spatial and temporal attention mechanisms and designs a gated fusion to fuse spatial and temporal information. ASTGNN [57] introduces a dynamic graph convolution module with a self-attention mechanism to capture spatial dependencies in a dynamic graph.

4.1.2 Temporal Dependencies

Urban spatial-temporal data exhibits temporal dependencies as a type of time-series data characterized by closeness, periodicity, and trend. For instance, a traffic jam at 8

am may influence the situation at 9 am, and rush hours may occur at similar times on consecutive workdays, repeating every 24 hours. Additionally, holidays and weekdays can also have a different impact on urban spatial-temporal data.

- Recurrent Neural Networks (RNNs): RNNs are a type of neural network designed to process sequential data and are thus well-suited for capturing the temporal dependencies in spatial-temporal data [15], [17], [37], [39], [46], [48], [64], [67]. To address the RNNs' inability to model long-term dependencies, LSTM [68] and GRU [69] are introduced as variants of the original RNN model. These models incorporate a gating mechanism that allows them to capture longer-term dependencies. The Seq2Seq model [70] is a popular framework that uses RNNs as both encoders and decoders to enable multi-step spatial-temporal prediction.
- Temporal Convolutional Networks (TCNs): TCNs are designed to process sequential data in a parallel fashion, which is impossible with Recurrent Neural Networks (RNNs) due to their reliance on historical information. TCNs are essentially 1-D CNN models that are used in the field of time series forecasting. They consider only the information before a given time step and are called causal convolutions. To increase the receptive field, dilated convolutions are used with causal convolutions. Multiple layers of dilated causal convolutions (also known as WaveNet [71]) can be stacked to achieve exponential growth of the receptive field. TCNs / 1D-CNN have become a common infrastructure in the field of spatial-temporal prediction [16], [45], [56], [61], [62], [72], [73].
- Temporal Attention Mechanisms: The temporal attention mechanism enables adaptive learning of nonlinear temporal dependencies in spatial-temporal data by assigning different weights to data at different time steps [38], [60], [61], [62], [63], [66], [73], [74]. This mechanism helps address the limitation of RNN models in modeling long-term dependencies, especially when combined with the Seq2Seq model. Recently, self-attention-based Transformer models have shown promising results in time series forecasting and have been applied to urban spatial-temporal prediction. For example, PDFormer [66] introduces a spatial-temporal self-attention mechanism to urban traffic flow prediction and incorporates the time-delay property of traffic state propagation.

4.1.3 Spatial-Temporal Dependencies Fusion

To capture the spatial-temporal dependencies in data, it is common to combine spatial and temporal models into a hybrid model, such as CNN+RNN, GCN+RNN, GCN+TCN, etc. The following are some common methods for fusing spatial and temporal models:

- Sequential Structure: The sequential structure combines the spatial and temporal neural networks in either parallel or serial configurations to form a spatial-temporal block [16], [39], [45], [56], [60], [61], [63], [67], [75]. For example, STGCN [16] utilizes 2 TCNs and 1 GCN to create a spatial-temporal block and stack them serially to capture the spatial-temporal dependencies in the data. TGCN [39] adopts a recursive approach by combining GCNs and GRUs, processing the input data recursively with GRUs after passing through GCNs at each time step. PDFormer [66] introduces a spatial-temporal self-attention-based model that employs multiple spatial self-attention heads and temporal self-attention heads in parallel to learn the spatial-temporal dependencies in the data.
- Coupled Structure: The coupled structure integrates spatial neural networks into the computation of temporal neural networks, most commonly by combining GCNs and RNNs [15], [46], [48]. This approach typically replaces the fully connected operations in the computation of RNNs, including its variants of GRU and LSTM models, with GCN operations. By doing so, spatial dependencies can be incorporated into the learning of temporal dependencies. Some examples of this approach include DCRNN [15], CCRNN [48], etc.
- Spatial-Temporal Synchronized Learning: Previous research on modeling spatial-temporal dependencies in data has often used separate components to independently learn spatial and temporal dependencies. However, a spatial-temporal synchronous learning approach can simultaneously directly capture local spatial-temporal dependencies in the data. To accomplish this, a spatial-temporal local graph structure is constructed across time steps, and GCN is used to learn spatial-temporal dependencies. STSGCN [41] and STFGCN [76] are two examples of this approach.

4.2 Micro Individual Prediction Tasks

Micro individual prediction tasks aim to incorporate spatial-temporal contextual information into individual behavior modeling. A trajectory $\mathcal{T} = [\langle \boldsymbol{v}_i, t_i \rangle]_{i=1}^{m}$ typically represents the moving object's route over time as a sequence of sample points $\boldsymbol{v}_i$. However, raw GPS trajectory data collected by IoT devices often have low and non-uniform sampling rates, and noisy points, which can be alleviated by discretizing the trajectory. The discretization process can involve map matching to obtain the *Road-network Constrained Trajectory* [77], [78], [79], mapping of latitude and longitude to the corresponding city region or grid to get the *Region or Grid-based Trajectory* [80], [81], [82] or considering the POI points with some attributes as the discrete trajectory unit, known as the *POI-based Trajectory* [26], [83], [84]. Once discretized, spatial-temporal context information can be modeled using deep learning methods in two steps [26], [78], [80], [84], described below. The first step is to model the basic units in the discretized trajectory using embedding methods. The second step is to model the trajectory sequence using sequence models.

4.2.1 Embedding Models

An embedding model is a fundamental method for converting discrete basic units of a trajectory into dense representation vectors. There are several types of embedding methods used in trajectory modeling:

- Feature Embedding: This method processes the features of the basic units, such as road length or POI category, to create a representation vector. Discrete features are

typically transformed into one-hot vectors, while continuous features may be normalized or bucketed. A fully connected layer is usually added on top of the processed representations to learn the unit representation.

- Graph Embedding: A trajectory's spatial basic units, such as a road network and POI graph, can be considered a graph structure. Graph embedding methods are used to learn the representation of trajectory units by incorporating the structural information of the graph [78], [80], [84], [85], [86]. This includes topology-aware methods, geospatial attributes-aware methods, and other techniques, which will be covered in more detail in Section 4.3.2 on road network representation learning.
- Spatial-temporal Multi-modal Embedding: Trajectories contain both spatial information and temporal and user characteristics. This method embeds the spatial-temporal features, and the user features into dense representations using multi-modal embedding modules. Spatial features use feature or graph embedding methods, while temporal features usually use embedding methods to learn the periodicity of the data. User features use feature embedding methods. Some studies propose more complex spatial-temporal embedding schemes for specific tasks. Different features are generally combined or summarized as a multi-modal representation [26], [78], [87], [88].

4.2.2 Sequence Models

Once the base units of the trajectory have been embedded into vector representations, the next step is to model the contextual information of the trajectory sequence. To achieve this, common sequence modeling techniques are applied, similar to Section 4.1.2.

- Recurrent Neural Networks (RNNs): RNNs are effective in dealing with sequential data and have been applied to learning spatial-temporal patterns in trajectory data [25], [26], [83], [89], [90], [91]. However, RNNs are prone to the vanishing gradient problem, which can be addressed using LSTM [68] or GRU [69], both incorporating gating mechanisms. Another popular RNN-based trajectory encoding model is the Seq2Seq [70] model, which learns trajectory representations through a trajectory reconstruction task [77], [80], [92].
- Attention Mechanisms: Attention mechanisms have been widely used in trajectory modeling to focus on essential parts of the trajectory. Combining the Seq2Seq model with the attention mechanism allows long-term dependencies to be better modeled. Recently, the Transformer structure, based on the self-attention mechanism, has been widely used in sequence data modeling. Many researchers have applied the Transformer module to model trajectory data with good results [78], [81], [86], [87], [88], [91], [93], [94], [95].

Apart from the generic modules described above, there are unique solutions for specific tasks, such as trajectory next-location prediction and travel time prediction, described separately in the following.

4.2.3 Trajectory Next-Location Prediction

Trajectory next-location prediction aims to predict the location that a user may visit next, given their historical trajectory. Two types of methods can be used based on the prediction target: probability-based and ranking-based models.

- Probability-based models: These models directly predict the probability distribution of the next location among all locations in the dataset and then determine the user's next location based on the predicted probability distribution [25], [26], [89], [90], [96]. For example, HST-LSTM [90] builds a spatial-temporal transfer matrix and integrates it into a standard LSTM network. This modified LSTM network, ST-LSTM, is used to build an encoder-decoder framework for predicting the next location.
- Ranking-based models: These models use negative sampling to obtain a candidate set to predict the location of the next hop. These models focus on ranking positive and negative samples and predicting the probability of each positive and negative sample being the next position [87], [91], [94], [95]. For example, STAN [87] applies the attention mechanism to encode the hidden state of the individual trajectory and designs another attention module for computing the dependency scores between the hidden state of the trajectory and the embedded positive and negative samples for predicting the rank score.

4.2.4 Travel Time Prediction

Travel time prediction aims to estimate the arrival time given the trajectory and departure time. There are three main methods: segment-based, trajectory-based, and OD-based.

- Segment-based methods: These methods estimate the travel time of each road segment and aggregate them to obtain the total travel time [97], [98]. PTTE [99] is an example that estimates segment travel times through tensor decomposition and determines the best concatenation of trajectories using dynamic programming. Although these methods are computationally efficient, they do not consider the sequence information of trajectories, such as the connections between road segments, which can result in error accumulation as the trajectory length increases.
- Trajectory-based methods: These methods estimate the total travel time end-to-end for the entire trajectory [78], [100], [101], [102], [103], [104], [105]. For instance, DeepTTE [105] uses geo-convolution to incorporate geographic information into traditional convolution, capturing spatial dependencies. TTPNet [102] leverages tensor decomposition and graph embedding to extract travel speed and road network representation effectively. START [78] proposes trajectory representation learning based on self-supervised pre-training for travel time prediction.
- OD-based methods: These methods estimate travel time for an origin-destination (OD) pair using only the departure time, origin, and destination [106]. For instance, Wang et al. [107] predict the travel time for OD pairs

using nearest-neighbor matching. NASF [108] combines the A^* algorithm with a neural network algorithm to estimate the fastest travel time between OD pairs.

4.3 Fundamental Tasks

Fundamental tasks provide support for macro and micro prediction tasks. We review the map matching and road network representation learning tasks separately.

4.3.1 Map Matching

Map matching aims to match the raw GPS trajectories to the road segments of the road network. The main methods can be divided into three categories:

- Rule-based methods: In the early stages, map matching methods used simple geometric or distance-based rules to match GPS points with the nearest road segments [109]. However, these rule-based methods have limitations due to their inability to handle GPS point errors and the complexity of the road network, which result in low matching accuracy.
- Probabilistic-based methods: Probabilistic-based map matching methods calculate the probability of matching GPS points to road segments. Commonly used probabilistic models include Hidden Markov Model (HMM) [110], [111], [112], Conditional Random Field (CRF) [113], and Weighted Graph Technique (WGT) [114], [115]. HMMs use probabilistic inference to find the most likely sequence of road segments based on GPS observations, accounting for uncertainties and errors. CRFs offer a more complex modeling approach by including features capturing various aspects of road network topology and geometry. WGT builds a weighted graph of possible vehicle routes and selects the most likely route based on GPS observations.
- Deep learning-based methods: Map matching methods based on deep learning have emerged in recent years, using neural networks to model the matching problem by learning features from large-scale trajectory data [116], [117], [118], [119]. One example is DeepMM [116], which performs map matching in the latent space using embedding techniques and a sequence learning model with attention enhancement. Another is DMM [117], which uses an encoder-decoder framework for a map matching model with variable-length input and output, and a reinforcement learning-based model for optimizing the matched outputs, specifically for map matching on cellular data.

4.3.2 Road Network Representation Learning

The aim of road network representation learning is to extract meaningful representations from raw road network data that can be used in downstream tasks such as trajectory mining [78], road segment classification [120], POI tour recommendation [84], and more. All types of graph relation data in Section 2.2 can be characterized for learning, including sensor graphs, POI graphs, region graphs, and grid graphs. However, road network representation learning is one of the most extensively studied topics in this field [31], [32], [79], [86], [120], [121], [122]. The main methods can be divided into three categories:

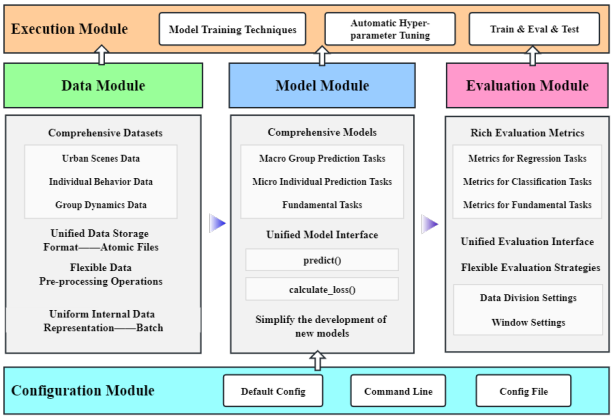


Fig. 2. Overview of the LibCity library

- Topology-aware graph embedding methods: Road network representation learning can be considered a graph representation or graph embedding technique. These methods apply topology-aware graph embedding techniques such as DeepWalk [123], Node2Vec [124], and LINE [125] to represent the road network as a graph. However, these methods tend to perform poorly in traffic-related tasks as they ignore the complex road properties and traffic semantics.
- Geospatial attributes-aware methods: To address the limitations of topology-aware methods, several methods have been proposed that extend graph representation learning techniques to road networks, such as RFN [31], IRN2Vec [32], RN2Vec [121] and HRNR [120]. Specifically, HRNR constructs a hierarchical neural architecture, *i.e.*, "road segments, structural regions, functional zones", to learn graph embeddings from different levels, focusing on both road geospatial attributes and graph topology. Later, researchers propose integrating trajectory features into road representation learning, such as Toast [79], which suggests that road networks are closely related to trajectories.
- Self-supervised learning methods: Recently, self-supervised learning techniques have gained traction in road network representation learning. JCLRNT [86] and SARN [122] are two notable methods that have applied self-supervised learning to this field. JCLRNT proposes a unified framework for jointly learning road network and trajectory representations using contrastive representation learning. On the other hand, SARN proposes a two-level graph contrastive loss function to guide the model in learning both the topology and spatial structure of road networks in a self-supervised manner. It is worth mentioning that self-supervised learning has also been utilized in trajectory representation learning [78], [88], [126], POI tour recommendation [83], [84], and other related areas.

5 The Library: LibCity

Deep learning algorithms have been extensively used in urban spatial-temporal prediction in recent years, resulting in a wealth of research findings. However, according to our survey, less than 30% of the papers published in 11

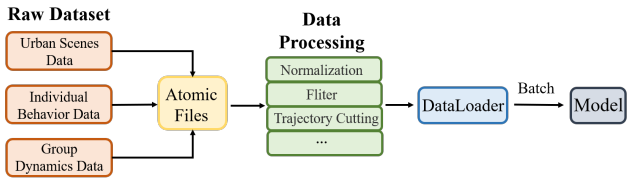


Fig. 3. The data processing flow in LibCity

leading conferences and journals have made their code and data open source, which hampers reproducibility in the field. Additionally, the need for recognized state-of-the-art (SOTA) models, standardized datasets, and transparent experimental settings creates obstacles to assessing the performance of new models and ultimately stifles innovation. In contrast, domains like Computer Vision, Natural Language Processing, and Recommendation Systems have standardized datasets such as IMAGENET [8] and algorithm libraries like MMDetection [9], and RecBole [10]. Unfortunately, urban spatial-temporal prediction lacks these resources. We propose LibCity, a comprehensive and unified library for urban spatial-temporal prediction, to address this issue.

Figure 2 illustrates the framework of LibCity, which comprises five main modules, including the data, model, evaluation, execution, and configuration module. Combining these modules forms a cohesive pipeline that provides researchers with a reliable experimental environment, and each module is responsible for a specific step in the pipeline. The subsequent sections will provide a detailed description of the implementation of each module.

- **Data Module**: Responsible for loading datasets and data preprocessing.
- **Model Module**: Responsible for initializing the reproduced baseline model or custom model.
- **Evaluation Module**: Responsible for evaluating model prediction results through multiple metrics.
- **Execution Module**: Responsible for model training and prediction.
- **Configuration Module**: Responsible for managing all the parameters involved in the framework.

5.1 Data Module

Our data module aims to address the issue of unfair evaluation caused by different data preparation methods by establishing a standardized data processing flow, as shown in Figure 3. This flow encompasses two types of data: user-oriented and model-oriented. The former establishes a unified storage format for spatial-temporal data, referred to as *atomic files*, which provides users with a consistent data input format. The latter defines a key-value data structure, known as *Batch*, to facilitate uniform data interaction between the data module and model module. Once the atomic files have been loaded and preprocessed, the *Dataloader* class in PyTorch is employed to convert the data into a *Batch* structure and then fed to the model module.

5.1.1 Atomic Files

LibCity defines a general and extensible data storage format for urban spatial-temporal data, *i.e.*, *atomic files*. Atomic files contain five categories, the minor information units in spatial-temporal data. For more details, please refer to Section 2.3 above.

5.1.2 Preprocessing Operations

The LibCity library provides support for various data preprocessing operations. For the macro group prediction task, it is essential to normalize the spatial-temporal data to improve the model's convergence toward the optimal solution. LibCity supports multiple data normalization methods, including Z-score normalization, min-max normalization, logarithmic normalization, and other custom normalization methods, which can be easily achieved through parameter settings. LibCity also supports the handling of missing values and outliers. For the micro individual prediction task, LibCity incorporates two trajectory filtering methods: inactive user filtering and inactive POI filtering. Inactive users can be filtered out by evaluating their activity based on the number and length of their trajectories and setting the minimum number of trajectories and minimum trajectory length. Similarly, inactive POIs can be filtered out based on the minimum number of visits. In this way, we can filter the data to reduce the impact of sparsity in spatial-temporal data.

5.1.3 Batch

The *Batch* is a key-value data structure based on the implementation of *python.dict*. It consists of feature names as keys and corresponding feature tensors (*torch.Tensor*) in a mini-batch as values. The purpose of the *Batch* is to facilitate data interaction between the data module and the model module in a systematic pipeline. In the pipeline, the executor extracts one *Batch* object from the *Dataloader* at a time and feeds it into the model. This structure enables the convenient use of feature tensors by referring to corresponding feature names. As different prediction models use varying features, they can all be stored in the form of *Batch*. Through this data form, LibCity can construct unified model interfaces and implement general executor for model training and testing, which is particularly useful for developing new models.

5.1.4 Comprehensive Datasets

We have conducted a comprehensive literature survey on spatial-temporal prediction and selected 351 representative papers, including survey papers. We identified all the open datasets used from these papers and selected 55 datasets based on their popularity, time span length, and data size. These datasets cover all the 9 tasks that LibCity supports, consisting of 40 group dynamics datasets (*SSTD data*), ten individual behavior datasets (*SDTD data*), and five urban scenes datasets (*road network data*). Table 2, Table 3, and Table 4 provide statistics on these datasets. To facilitate the use of these datasets in LibCity, we have converted all of them into the atomic file format and created conversion tools, which can be found at this link[2].

2. https://github.com/LibCity/Bigscity-LibCity-Datasets

TABLE 2
Summary of Group Dynamics Datasets (*SSTD Data*) in LibCity

DATASET	#GEO	#REL	#DYNA	PLACE	DURATION	#TS	DATA TYPE
METR-LA [15]	207	11,753	7,094,304	Los Angeles, USA	Mar. 1, 2012 - Jun. 27, 2012	5min	Graph Speed
Los-Loop [39]	207	42,849	7,094,304	Los Angeles, USA	Mar. 1, 2012 - Jun. 27, 2012	5min	Graph Speed
SZ-Taxi [39]	156	24,336	464,256	Shenzhen, China	Jan. 1, 2015 - Jan. 31, 2015	15min	Graph Speed
Q-Traffic [127]	45,148	63,422	264,386,688	Beijing, China	Apr. 1, 2017 - May 31, 2017	15min	Graph Speed
Loop Seattle [67], [128]	323	104,329	33,953,760	Greater Seattle Area, USA	Jan. 1, 2015 - Dec. 31, 2015	5min	Graph Speed
PEMSD7(M) [16]	228	51,984	2,889,216	California, USA	Weekdays of May. Jun., 2012	5min	Graph Speed
PEMS-BAY [15]	325	8,358	16,937,700	San Francisco Bay Area, USA	Jan. 1, 2017 - Jun. 30, 2017	5min	Graph Speed
Rotterdam [129]	208	-	4,813,536	Rotterdam, Holland	135 days of 2018	2min	Graph Speed
PeMSD3 [41]	358	547	9,382,464	California, USA	Sept. 1, 2018 - Nov. 30, 2018	5min	Graph Flow
PEMSD7 [41]	883	866	24,921,792	California, USA	Jul. 1, 2016 - Aug. 31, 2016	5min	Graph Flow
Beijing subway [130]	276	76,176	248,400	Beijing, China	Feb. 29, 2016 - Apr. 3, 2016	30min	Graph Flow
M-dense [131]	30	-	525,600	Madrid, Spain	Jan. 1, 2018 - Dec. 21, 2019	60min	Graph Flow
SHMetro [132]	288	82,944	1,934,208	Shanghai, China	Jul. 1, 2016 - Sept. 30, 2016	15min	Graph Flow
HZMetro [132]	80	6,400	146,000	Hangzhou, China	Jan. 1, 2019 - Jan. 25, 2019	15min	Graph Flow
NYCTaxi-Dyna[3]	263	69,169	574,392	New York, USA	Jan. 1, 2020 - Mar. 31, 2020	60min	Region Flow
PeMSD4 [61]	307	340	5,216,544	San Francisco Bay Area, USA	Jan. 1, 2018 - Feb. 28, 2018	5min	Graph Flow, Speed, Occupancy
PEMSD8 [61]	170	277	3,035,520	San Bernardino Area, USA	Jul. 1, 2016 - Aug. 31, 2016	5min	Graph Flow, Speed, Occupancy
TaxiBJ2013 [14]	32*32	-	4,964,352	Beijing, China	Jul. 1, 2013 - Oct. 30, 2013	30min	Grid In&Out Flow
TaxiBJ2014 [14]	32*32	-	4,472,832	Beijing, China	Mar. 1, 2014 - Jun. 30, 2014	30min	Grid In&Out Flow
TaxiBJ2015 [14]	32*32	-	5,652,480	Beijing, China	Mar. 1, 2015 - Jun. 30, 2015	30min	Grid In&Out Flow
TaxiBJ2016 [14]	32*32	-	6,782,976	Beijing, China	Nov. 1, 2015 - Apr. 10, 2016	30min	Grid In&Out Flow
T-Drive [133], [134]	32*32	-	3,686,400	Beijing, China	Feb. 1, 2015 - Jun. 30, 2015	60min	Grid In&Out Flow
Porto[3]	20*10	-	441,600	Porto, Portugal	Jul. 1, 2013 - Sept. 30, 2013	60min	Grid In&Out Flow
NYCTaxi140103[4]	10*20	-	432,000	New York, USA	Jan. 1, 2014 - Mar. 31, 2014	60min	Grid In&Out Flow
NYCTaxi140112 [37]	15*5	-	1,314,000	New York, USA	Jan. 1, 2014 - Dec. 31, 2014	30min	Grid In&Out Flow
NYCTaxi150103 [135]	10*20	-	576,000	New York, USA	Jan. 1, 2015 - Mar. 1, 2015	30min	Grid In&Out Flow
NYCTaxi160102 [38]	16*12	-	552,960	New York, USA	Jan. 1, 2016 - Feb. 29, 2016	30min	Grid In&Out Flow
NYCBike140409 [14]	16*8	-	562,176	New York, USA	Apr. 1, 2014 - Sept. 30, 2014	60min	Grid In&Out Flow
NYCBike160708 [135]	10*20	-	576,000	New York, USA	Jul. 1, 2016 - Aug. 29, 2016	30min	Grid In&Out Flow
NYCBike160809 [38]	14*8	-	322,560	New York, USA	Aug. 1, 2016 - Sept. 29, 2016	30min	Grid In&Out Flow
NYCBike200709[5]	10*20	-	441,600	New York, USA	Jul. 1, 2020 - Sept. 30, 2020	60min	Grid In&Out Flow
AustinRide[6]	16*8	-	282,624	Austin, USA	Jul. 1, 2016 - Sept. 30, 2016	60min	Grid In&Out Flow
BikeDC[7]	16*8	-	282,624	Washington, USA	Jul. 1, 2020 - Sept. 30, 2020	60min	Grid In&Out Flow
BikeCHI[8]	15*18	-	596,160	Chicago, USA	Jul. 1, 2020 - Sept. 30, 2020	60min	Grid In&Out Flow
NYCTaxi-OD[4]	263	69,169	150,995,927	New York, USA	Apr. 1, 2020 - Jun. 30, 2020	60min	OD Flow
NYC-TOD [136]	15*5	-	98,550,000	New York, USA	Jan. 1, 2014 - Dec. 31, 2014	30min	Grid-OD Flow
NYCTaxi150103 [135]	10*20	-	115,200,000	New York, USA	Jan. 1, 2015 - Mar. 1, 2015	30min	Grid-OD Flow
NYCBike160708 [135]	10*20	-	115,200,000	New York, USA	Jul. 1, 2016 - Aug. 29, 2016	30min	Grid-OD Flow
NYC-Risk [137]	243	59,049	3,504,000	New York, USA	Jan. 1, 2013 - Dec. 31, 2013	60min	Risk
CHI-Risk [137]	243	59,049	3,504,000	New York, USA	Jan. 1, 2013 - Dec. 31, 2013	60min	Risk

5.2 Model Module

To increase the modularity of the library and reduce coupling between different modules, LibCity uses a separate model module to implement classic spatial-temporal prediction algorithms. This module contains various models such as LSTM, GRU, TCN, GCN, etc. Encapsulating each model in a separate class, LibCity enables users to easily switch between different models and extend the library with new ones.

5.2.1 Unified Interface

In specific, LibCity provides two standard interfaces for all urban spatial-temporal prediction models: *predict()* and *calculate_loss()* functions as follows:

- The *predict()* function is used in the process of model prediction to return the model prediction results.
- The *calculate_loss()* function is used in the model training process to return the loss value, which needs to be optimized.

Both methods take the internal data representation *Batch* as input. These interface functions are general to different spatial-temporal prediction models, which allows researchers to implement various models in a highly unified way. When developing a new model, researchers only need to instantiate these two interfaces to connect with other modules in LibCity. They do not need to worry about how each of the other parts works. This design simplifies the development process and accelerates the development of new models.

5.2.2 Implemented Models

Currently, LibCity supports 9 mainstream spatial-temporal prediction tasks. Through careful investigation of the development process in the field of spatial-temporal prediction, we have selected 65 classic spatial-temporal prediction models to reproduce, ranging from early CNN-based models to recent GCN-based models and hybrid models. Moreover, in order to cover a wide range of prediction models, we also implement four shallow baseline models. We have tested all implemented models' performance on at least two datasets. We summarize the implemented 65 models in Table 5. Referring to Section 4.1, we also provide in the table the different basic structures of the model in the spatial and temporal dimensions. Users can refer to this table to learn about the main techniques and developments in the field of urban spatial-temporal prediction.

5.3 Evaluation Module

With the standardized data processing flow and prediction model interfaces, LibCity also offers standard evaluation procedures for spatial-temporal prediction tasks. Since the model output formats and evaluation metrics may vary across different spatial-temporal prediction tasks, LibCity develops specific evaluators for each task and supports various popular evaluation metrics.

5.3.1 Evaluation Metrics

Metrics for Regression Tasks: In LibCity, regression tasks consist of Traffic Flow Prediction, Traffic Speed Prediction, On-Demand Service Prediction, Origin-Destination Matrix Prediction, Traffic Accidents Prediction, and Travel Time Prediction. These tasks output real numbers and are evaluated using commonly used value-based metrics, which include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), Coefficient of Determination (R^2), and Explained Variance Score (EVAR). Their calculation formulas for these metrics are as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{3}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{4}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{5}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}|\frac{\hat{y}_i - y_i}{y_i}| * 100\% \tag{6}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{7}$$

$$EVAR = 1 - \frac{Var(y_i - \hat{y}_i)}{Var(y_i)} \tag{8}$$

where $\boldsymbol{y} = \{y_1, y_2, ..., y_n\}$ is the ground-truth value, $\hat{\boldsymbol{y}} = \{\hat{y_1}, \hat{y_2}, ..., \hat{y_n}\}$ is the prediction value, n is the number of samples, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the mean value, $Var(y_i) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$ is the variance.

Metrics for Classification Tasks: The classification task in LibCity is Trajectory Next-Location Prediction. The output of this task in LibCity is a probability distribution over the candidate's next locations. The Trajectory Next-Location Prediction task is evaluated using various rank-based metrics, including Precision@K, Recall@K, F1-score@K, MRR@K (Mean Reciprocal Rank@K), and NDCG@K (Normalized Discounted Cumulative Gain@K). The calculation formulas for these metrics are as follows:

$$Precision@K = \frac{\sum_{i=1}^{N}|\,\mathrm{Hit}(i)|}{N \times K} \tag{9}$$

$$Recall@K = \frac{\sum_{i=1}^{N}|\,\mathrm{Hit}(i)|}{N} \tag{10}$$

$$F1@K = \frac{2 \times \text{ Precision@ } \times \text{ Recall@ } K}{\text{Precision } @ + \text{ Recall@ } K} \tag{11}$$

$$MRR@K = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\mathrm{Rank}(i)} \tag{12}$$

$$NDCG@K = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\log_2(\mathrm{rank}(i)+1)} \tag{13}$$

3. https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i
4. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
5. https://www.citibikenyc.com/system-data
6. https://data.world/ride-austin/ride-austin-june-6-april-13
7. https://www.capitalbikeshare.com/system-data
8. https://www.divvybikes.com/system-data
9. https://www.openstreetmap.org

TABLE 3
Summary of Individual Behavior Datasets (*SDTD Data*) in LibCity

DATASET	#GEO	#REL	#USR	#DYNA	PLACE	DURATION	DATA TYPE
Seattle [110]	613,645	857,406	1	7,531	Seattle WA, USA	Jan. 17, 2009	GPS-based
Global [138]	11,045	18,196	1	2,502	100 cities	-	GPS-based
CD-Taxi-Sample [105]	-	-	4,565	712,360	Chengdu, China	Aug. 3, 2014 - Aug. 30, 2014	GPS-based
BJ-Taxi-Sample [105]	16,384	-	76	518,424	Beijing, China	Oct. 1, 2013 - Oct. 31, 2013	GPS-based
Porto [78]	10,903	26,161	435	695,085	Porto, Portugal	Jul. 1, 2013 - Jul. 1, 2014	Road-network Constrained
Foursquare-TKY [139]	61,857	-	2,292	573,703	Tokyo, Japan	Apr. 4, 2012 - Feb. 16, 2013	POI-based
Foursquare-NYC [139]	38,332	-	1,082	227,428	New York, USA	Apr. 3, 2012 - Feb. 15, 2013	POI-based
Gowalla [140]	1,280,969	913,660	107,092	6,442,892	Global	Feb. 4, 2009 - Oct. 23, 2010	POI-based
BrightKite [140]	772,966	394,334	51,406	4,747,287	Global	Mar. 21, 2008 - Oct. 18, 2010	POI-based
Instagram [141]	13,187	-	78,233	2,205,794	New York, USA	Jun. 15, 2011 - Nov. 8, 2016	POI-based

TABLE 4
Summary of Urban Scenes Datasets (Road Network Data) in LibCity

DATASET	#GEO	#REL	PLACE
BJ-Roadmap-Edge [78]	40,306	101,024	Beijing, China
BJ-Roadmap-Node[9]	16,927	38,027	Beijing, China
CD-Roadmap-Edge[9]	6,195	15,962	Chengdu, China
XA-Roadmap-Edge[9]	5,269	13,032	Xian, China
Porto-Roadmap-Edge [78]	11,095	26,161	Porto, Portugal

where N is the number of test data, i is the i-th test data, K is the top K prediction outputs for evaluation, $T(i)$ is the real next hop position in the i-th test data, $R(i)$ is the set of the top K locations in the prediction result of the i-th test data, $Hit(i)$ is the set of predicted hit locations in the i-th test data, which means $T(i) \cap R(i)$, $Rank(i)$ is the ranking of $T(i)$ in $R(i)$ in the i-th test data, and $|*|$ is the modulo operator of a set.

Metrics for Fundamental Tasks: Fundamental tasks in LibCity provide support for macro and micro prediction tasks, including map matching and road network representation learning. The road network representation task requires combining with specific downstream tasks to evaluate the performance of the representation vectors. For instance, if a road segment classification task is used for evaluation, it is a classification task. If a road flow prediction task is used for evaluation, it is a regression task. The evaluation metrics for these tasks are similar to the ones described above. Here we focus on the evaluation metrics for the map matching task.

LibCity evaluates the map matching task using three metrics: RMF (Route Mismatch Fraction), AN (Accuracy in Number), and AL (Accuracy in Length), which have been used in previous works such as [110], [154].

$$RMF = \frac{d_- + d_+}{d_0} \tag{14}$$

$$AN = \frac{\#Rc}{\#R} \tag{15}$$

$$AL = \frac{\sum len(Rc)}{\sum len(R)} \tag{16}$$

where d_0 denotes the length subtracted from the error, d_+ denotes the length added to the error, d_- is the total length of the real path, Rc denotes correctly matched roads, R denotes all roads of the real route. $len()$ denotes the length of a set of roads.

5.3.2 Evaluation Strategies

To evaluate the performance of spatial-temporal prediction models in a flexible manner, LibCity offers two main strategies.

Firstly, users can divide the dataset into training, validation, and testing sets with a ratio of their choice. The training set is used for model training, while the validation set is used for hyper-parameter tuning and preventing overfitting. The testing set is used to evaluate the final performance of the trained model.

Secondly, LibCity allows users to set different window sizes for evaluation. For macro group prediction tasks, users can set various input and output time windows, and LibCity will partition the input data based on the window size, enabling multi-step predictions using historical observations of different lengths. For micro individual prediction tasks, trajectories are split based on window settings, with options for time-based or length-based windows. Users can set the window size and type to evaluate the model's performance on trajectories of varying lengths, such as long, medium, and short trajectories.

These two strategies can be combined with various evaluation metrics to perform comprehensive evaluations of models for the same task, providing greater flexibility and adaptability in assessing model performance.

5.4 Execution Module

The execution module in LibCity serves as the central hub that controls the interactions between other modules to facilitate model training and performance evaluation. Users can modify the parameter settings of this module to adjust the effect of model training. It supports various model training strategies that optimize the model and includes a built-in automatic hyper-parameter tuning module to reduce the user's workload and achieve automatic optimization of the model.

TABLE 5
Implement Models in LibCity

Task	Model	Conference	Year	Spatial Axis			Temporal axis			
				CNN	GCN	Attn.	LSTM	GRU	TCN	Attn.
Traditional Methods	HA	-	-	-	-	-	-	-	-	-
	SVR [35]	NIPS	1996	-	-	-	-	-	-	-
	ARIMA [34]	J TRANSP ENG	2003	-	-	-	-	-	-	-
	VAR [33]	Princeton Press	1994	-	-	-	-	-	-	-
General Macro Group Prediction	RNN [70]	NIPS	2014				✓	✓		
	Seq2Seq [70]	NIPS	2014				✓	✓		
	AutoEncoder [142]	IEEE TITS	2014	-	-	-	-	-	-	-
	FNN [15]	ICLR	2018	-	-	-	-	-	-	-
Traffic Flow Prediction	ST-ResNet [14]	AAAI	2017	✓						
	STNN [143]	ICDM	2017	-	-	-	-	-	-	-
	ACFM [37]	ACM MM	2018	✓		✓	✓			
	STDN [135]	AAAI	2019	✓			✓			✓
	ASTGCN [61]	AAAI	2019		✓	✓			✓	✓
	MSTGCN [61]	AAAI	2019		✓				✓	
	DSAN [38]	KDD	2020			✓				✓
	STSGCN [41]	AAAI	2020		✓					
	AGCRN [46]	NIPS	2020		✓			✓		
	CRANN [144]	arXiv	2020	✓		✓	✓			
	CONVGCN [145]	IET ITS	2020	✓	✓					
	ResLSTM [130]	IEEE TITS	2020		✓		✓			✓
	MultiSTGCnet [146]	IJCNN	2020		✓		✓			
	ToGCN [75]	IEEE TITS	2021		✓		✓			
	DGCN [73]	IEEE TITS	2022		✓	✓	✓		✓	✓
Traffic Speed Prediction	DCRNN [15]	ICLR	2018		✓			✓		
	STGCN [16]	IJCAI	2018		✓				✓	
	GWNET [56]	IJCAI	2019		✓				✓	
	TGCN [39]	IEEE TITS	2019		✓			✓		
	TGCLSTM [67]	IEEE TITS	2019		✓		✓			
	MTGNN [45]	KDD	2020		✓				✓	
	GMAN [60]	AAAI	2020			✓				✓
	ATDM [147]	NCA	2021		✓		✓			
	STAGGCN [62]	CIKM	2020		✓	✓			✓	✓
	ST-MGAT [148]	ICTAI	2020			✓			✓	
	DKFN [149]	ACM GIS	2020		✓		✓			
	STTN [63]	arXiv	2020		✓	✓				✓
	HGCN [72]	AAAI	2021		✓				✓	✓
	GTS [150]	ICLR	2021		✓			✓		
On-Demand Service Prediction	DMVSTNet [17]	AAAA	2018	✓			✓			
	STG2Seq [74]	IJCAI	2019		✓					✓
	CCRNN [48]	AAAI	2021		✓			✓		
Traffic Accident Prediction	GSNet [137]	AAAI	2021		✓			✓		✓
OD Matrix Prediction	GEML [151]	KDD	2019		✓		✓			
	CSTN [136]	IEEE TITS	2019	✓			✓			
Trajectory Next-Location Prediction	FPMC [152]	WWW	2010							
	LSTM [153]	CoRR	2013				✓			
	ST-RNN [25]	AAAI	2016					✓		
	SERM [89]	CIKM	2017					✓		
	DeepMove [26]	WWW	2017				✓			✓
	CARA [95]	SIGIR	2018					✓		
	HSTLSTM [90]	IJCAI	2018				✓			
	ATSTLSTM [91]	IEEE TSC	2019				✓			✓
	LSTPM [96]	AAAI	2020					✓		✓
	GeoSAN [94]	KDD	2020							✓
	STAN [87]	WWW	2021							✓
Travel Time Prediction	DeepTTE [105]	AAAI	2018	✓			✓			
	TTPNet [102]	TKDE	2022	✓			✓			
Map Matching	STMatching [115]	ACM GIS	2009	-	-	-	-	-	-	-
	HMMM [110]	ACM GIS	2009	-	-	-	-	-	-	-
	IVMM [154]	IEEE MDM	2010	-	-	-	-	-	-	-
Road Network Representation Learning	DeepWalk [123]	KDD	2014	-	-	-	-	-	-	-
	LINE [125]	WWW	2015	-	-	-	-	-	-	-
	Node2Vec [124]	KDD	2016	-	-	-	-	-	-	-
	ChebConv [55]	NIPS	2016		✓					
	GAT [155]	arXiv	2017			✓				
	GeomGCN [156]	ICLR	2020		✓					

5.4.1 Model Training Techniques

LibCity offers various training techniques to train deep neural networks effectively. These techniques can be customized by modifying the parameter settings of the execution module. Here are some of the techniques supported by LibCity:

- *Optimizer options*: During the training of a deep learning model, the optimizer is mainly used to update the parameters of the network in order to minimize the loss function. LibCity supports several optimizers including SGD [157], RMSProp [158], Adam [159], and AdaGrad [160]. Different optimization algorithms have different ways of updating the network parameters and are better suited to different scenarios. Users can choose the appropriate optimization algorithm for their usage scenarios to achieve better training results.
- *Learning rate adjustment strategies*: Learning rate is a key parameter of neural network models, which controls the speed of gradient-based adjustment of network weights and determines the convergence of the loss function to the optimal solution.
- *Loss Function options*: Various loss functions are used in deep learning and different loss functions compute the loss in different ways to obtain different training results. LibCity supports five types of loss functions including Cross-entropy Loss, L1 Loss, L2 Loss, Huber Loss [161], LogCosh Loss [162], and Quantile Loss [163].
- *Early Stopping*: Overfitting may occur as the number of training rounds increases. LibCity supports the early stopping method to prevent overfitting. Users can specify whether to use the early stop mechanism and the size of the duration rounds through parameter configuration.
- *Gradient Clipping*: Gradient explosion may occur during training. To avoid this, LibCity supports gradient clipping strategies [164]. Users can specify whether to use gradient clipping through parameter configuration.

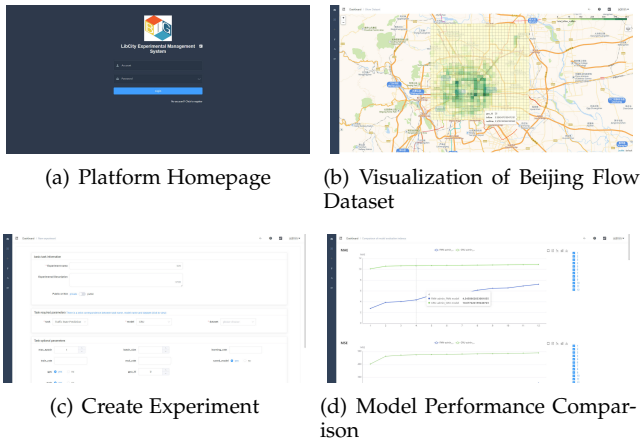
5.4.2 Automatic Hyper-parameter Tuning

Hyper-parameter tuning has a significant impact on the performance of deep learning models. To ease the burden of manual parameter tuning, LibCity provides an automatic hyper-parameter tuning mechanism. We implement this feature using the third-party library *Ray Tune* [165], which supports various search algorithms such as Grid Search, Random Search, and Bayesian Optimization. Users can specify the parameters to be tuned and their search space in a configuration file and select the tuning method. LibCity will then sample multiple times from the search space and run the model in a distributed manner, automatically saving the best parameter values and corresponding model prediction results. An example of automatic hyper-parameter tuning is presented in Section 6.2.

5.5 Configuration Module

LibCity utilizes the configuration module to set the parameters for the entire framework. The experiment parameter configuration is determined by three factors: parameters passed from the command line, user-defined configuration file, and default configuration files of the modules in LibCity. The priority of the above three parameter configuration methods decreases in order, with higher priority parameters overriding lower priority parameters with the same name. With this priority design, users can flexibly adjust the parameter configuration of an experiment through the first two methods.

To adjust parameters through the command line, users only need to use "−−parameter_name" when running LibCity. However, it is important to note that only frequently adjusted parameters in an experiment, such as batch size and learning rate, are allowed to be passed from the command line. In order to allow users to modify default parameters more extensively, LibCity allows users to pass the name of a user-defined configuration file through the command line, which is then read by the system to set the parameter configuration.



(a) Platform Homepage (b) Visualization of Beijing Flow Dataset

(c) Create Experiment (d) Model Performance Comparison

Fig. 4. Experiment Management and Visualization Platform

5.6 Experiment Management and Visualization Platform

We have created a web-based experiment management and visualization platform for convenient experimentation with LibCity's models and datasets as shown in Figure 4(a). The platform features a user-friendly graphical interface and comprehensive functionalities to support spatial-temporal prediction research. Users can easily upload new datasets and visualize them, as demonstrated in Figure 4(b), which shows the visualization of the Beijing traffic flow dataset. After configuring the model's parameters, users can create new experiments through a straightforward web interface, as shown in Figure 4(c). They can then execute the experiments at their convenience, with the training logs available for viewing during the experiment. After the experiment's execution, users can obtain evaluation results of the model on specific metrics and visualize the prediction results. Additionally, the platform offers an experimental comparison function, allowing users to compare the performance of different models shown in Figure 4(d).

We have built the experiment management and visualization platform using Django[10], Vue[11], and MySQL[12]. The

platform's open-sourced code can be found on GitHub[13].

6 Usage Examples of LibCity

This section provides several examples of using LibCity to assist users in getting started with the framework. The examples cover running existing models, conducting automatic parameter tuning, and adding new models to LibCity.

6.1 Running Existing Models

The general process of running existing models in LibCity is listed as follows:

i) Dataset formatting: Users must download and convert the raw dataset into atomic files.
ii) Configuration setup: Users can configure the experiment parameters through a configuration file, the command line, or the default parameters of LibCity. The configuration module's parameters form the foundation of the entire framework.
iii) Dataset pre-processing and splitting: Based on the user's parameter settings, LibCity pre-processes and splits the dataset. Users can specify custom dataset division ratios and data pre-processing thresholds. For example, users may filter out trajectories with lengths less than 5.
iv) Model initialization: LibCity creates a model object based on the user's task and model name selection.
v) Training and evaluation: Once the data and model are ready, LibCity calls the executor module to train and evaluate the model on the specified dataset. The training logs, trained models, and model evaluation results are automatically saved for the user's use.

The above steps are performed automatically by LibCity's unified entry file (*run_model.py*). Users need only execute a single command in the command line to initiate the model running process. Three essential parameters must be specified at runtime, namely the task, model, and dataset, via *--task*, *--model*, and *--dataset* options, respectively. For example, running the GRU model on the METR_LA dataset for 50 epochs on the second GPU block requires the following command:

python run_model.py --task traffic_state_prediction --model GRU --dataset METR_LA --gpu 2 --epoch 50

6.2 Running Automatic Parameter Tuning

Considering that hyper-parameter tuning significantly impacts the performance of deep learning models, LibCity has introduced an automatic hyper-parameter tuning mechanism that can easily optimize a given model based on the user-provided hyper-parameter range. The general steps for running the automatic tuning function are as follows:

i) Setting the hyper-parameter search space. Users must specify the hyper-parameters to be tuned and their corresponding value ranges in a JSON file. For example, the search space for the hidden layer dimension parameter can be a discrete set of categorical variables, such as [50, 100, 200]. The user can specify the file name of the parameter space file using the parameter *--space_file*.
ii) Selecting the tuning method. LibCity uses the third-party library *Ray Tune* [165] to implement automatic hyper-parameter tuning, which supports various search algorithms, including Grid Search, Random Search, and Bayesian Optimization. The user can select the tuning method by specifying the parameter *--search_alg*.
iii) Starting the tuning process. Users can execute a single command in the command line to automatically tune the model parameters. During the tuning process, LibCity will sample the corresponding parameter values from the search space and perform model training and validation. After all the samples are validated, the script will output the best parameter combination on the terminal and save them to a log file. Here is an example of the command to run the automatic tuning function: *python hyper_tune.py --task traffic_state_pred --model GRU --dataset METR_LA --search_alg BasicSearch --space_file sample_space_file*

6.3 Implementing a New Model

The modular design and unified interface definition of LibCity provide flexibility for user-defined extensions and excellent scalability. Developing a new spatial-temporal prediction model using LibCity is straightforward. The general process for developing a new model using LibCity is as follows:

i) Create a new model file and define a model class that inherits from the existing model abstraction class provided by LibCity. All 9 tasks supported by LibCity have their abstract classes implemented.
ii) Implement the *__init__()* function to initialize the model according to the configuration parameters. The input parameter *config* contains the configuration information.
iii) Implement the *calculate_loss()* function to calculate the loss between the predicted result and the true value during model training. The goal of model training is to optimize this loss.
iv) Implement the *predict()* function to return the model prediction results during prediction.
v) Configure the default parameter configuration file for the new model to specify the required parameters and their values for running the model.
vi) Run the model and evaluate its performance on the selected dataset.

Thus, developing new models using LibCity is simplified by focusing on implementing only three interfaces, while LibCity handles other details like data splitting, model training, and performance evaluation. This design approach simplifies the development process of new models and highlights the scalability of LibCity.

7 Spatial-Temporal Prediction Benchmark

We have conducted comprehensive experiments on various models and datasets to set up a benchmark for the spatial-temporal prediction field.

10. https://www.djangoproject.com
11. https://vuejs.org
12. https://www.mysql.com
13. https://github.com/LibCity/Bigscity-LibCity-WebTool

7.1 Benchmarks on Macro Group Prediction Tasks

The Macro Group Prediction Tasks involve predicting the spatial-temporal attributes of macro groups in the future based on historical observations of Spatial Static Temporal Dynamic (SSTD) data. Since the technical development lines for different macro group prediction tasks are similar, only the datasets' attributes vary. We conduct a unified experiment on different datasets to compare the performance of different model structures.

7.1.1 Datasets

Here we consider the following 19 datasets, which can be categorized into two types of data: Graph Relation Data and Grid Relation Data. The datasets are listed below, and those marked with an asterisk (*) in the upper right corner are new datasets released in this study:

(1) Graph Relation Data: Traffic speed or flow data recorded by traffic sensors or based on urban trajectories.

- METR-LA [15]: This dataset describes the highway speed of Los Angeles County from Mar. to Jun. 2012.
- PEMS-BAY [15]: This dataset describes the highway speed of San Francisco Bay from Jan. to May 2017.
- PEMSD7(M) [16]: This dataset describes the highway speed of California on the weekdays of May and Jun. of 2012.
- PEMSD4-Speed [61]: This dataset describes the speed of California freeway from Jan. to Feb. 2018.
- PEMSD8-Speed [61]: This dataset describes the speed of California freeways from Jul. to Aug. 2016.
- PEMSD4-Flow [61]: This dataset describes the flow of California freeway from Jan. to Feb. 2018.
- PEMSD8-Flow [61]: This dataset describes the flow of California freeways from Jul. to Aug. 2016.
- PEMSD4-Occupy [61]: This dataset describes the traffic occupancy of California freeway from Jan. to Feb. 2018.
- PEMSD8-Occupy [61]: This dataset describes the traffic occupancy of California freeways from Jul. to Aug. 2016.

(2) Grid Relation Data: City grid traffic inflow and outflow data based on the urban taxi or bike trajectories.

- NYCTaxi150103 [135]: New York taxi flow, the grid size is 10*20, and the time range is Jan. 1 - Mar. 1, 2015.
- NYCTaxi160102 [38]: New York taxi flow, the grid size is 16*12, and the time range is Jan. 1 - Feb. 29, 2016.
- NYCTaxi140103*: New York taxi flow, the grid size is 10*20, and the time range is Jan. 1 - Mar. 31, 2014.
- TaxiBJ2014 [14]: Beijing taxi flow, the grid size is 32*32, and the time range is Mar. 1 - Jun. 30, 2014.
- TaxiBJ2015 [14]: Beijing taxi flow, the grid size is 32*32, and the time range is Mar. 1 - Jun. 30, 2015.
- NYCBike160708 [135]: New York bike flow, the grid size is 10*20, and the time range is Jul. 1- Aug. 29, 2016.
- NYCBike140409 [14]: New York bike flow, the grid size is 16*8, and the time range is Apr. 1 - Sept. 29, 2014.
- NYCBike200709*: New York bike flow, the grid size is 10*20, and the time range is Jul. 1 - Sept. 30, 2020.
- BikeDC*: Washington bike flow, the grid size is 16*8, and the time range is Jul. 1 - Sept. 30, 2020.
- BikeCHI*: Chicago bike flow, the grid size is 15*18, and the time range is Jul. 1 - Sept. 30, 2020.

7.1.2 Models

We conducted experiments on the following models that can be divided into six categories as follows:

(1) General Macro Group Prediction Baseline Methods:

- Seq2Seq [70]: Sequence-to-sequence model, we utilize the encode-decoder framework based on the gated recurrent unit for multi-step prediction.
- AutoEncoder [142]: AutoEncoder uses an encoder to learn an embedded vector from data and then uses a decoder to predict the future state.
- FNN [15]: Feed forward neural network with two hidden layers and L2 regularization.

(2) Sequential Structure (Spatial CNN plus others)

- STResNet [14] models the spatial-temporal correlations by residual unit.
- ACFM [37]: infers the evolution of the crowd flow with a ConvLSTM and attention mechanism
- DMVSTNet [17] combines CNN and LSTM to capture the spatial, temporal, and semantic views.

(3) Sequential Structure (Spatial GNN plus others)

- STGCN [16] combines graph convolutions and gated temporal convolution.
- GWNET [56] combines adaptive adjacency matrix into graph convolutional network and 1D dilated casual convolution.
- MTGNN [45] combines adaptive graph convolutional network with mix-hop propagation layer and dilated inception layer.
- TGCN [39] combines graph convolutional network and gated recurrent unit.
- TGCLSTM [67] combines the K-hop graph convolutional network and long short-term memory network.

(4) Sequential Structure (Spatial Attn. plus others)

- ASTGCN [61] pluses the STGCN with spatial-temporal attention mechanism.
- GMAN [60] consists of multiple spatial-temporal attention blocks in the encoder-decoder framework.
- STTN [63] combines the spatial and temporal transformer and graph convolutional network.

(5) Coupled Structure (GNN + RNN):

- DCRNN [15]: couples the diffusion graph convolution and the encoder-decoder framework.
- AGCRN [46]: couples the adaptive graph convolution and the gated recurrent unit.

(6) Spatial-Temporal Synchronized Learning:

- STG2Seq [74] utilizes a hierarchical graph convolutional to capture spatial and temporal correlations simultaneously.
- STSGCN [41] utilizes spatial-temporal synchronous modeling mechanism to model localized correlations.

7.1.3 Experimental Settings

- Data Division Settings: We split all datasets into the training, validation, and test sets with a ratio of 7:1:2.
- Prediction Window Settings: For Graph Relation Data, we performed multi-step prediction using 12 past steps to predict 12 future steps. For Grid Relation Data, we performed single-step prediction using six past steps to predict the next single-step spatial-temporal attributes.

TABLE 6
Baseline Model Performance Ranking for Macro Group Prediction Tasks

1	2	3	4	5	6
MTGNN	GWNET	AGCRN	GMAN	DCRNN	STGCN
7	8	9	10	11	12
STTN	STSGCN	STG2Seq	ASTGCN	TGCN	TGCLSTM
13	14	15	16	17	18
DMVSTNet	ACFM	STResNet	Seq2Seq	FNN	AutoEncoder

- Evaluation Metrics: We used three widely-used metrics, MAE, RMSE, and MAPE, to evaluate the models. Missing values were excluded from training and testing following [15], and samples with flow values below five were filtered out when testing the models on grid relation data following [17]. We repeated all experiments five times and reported the average results. For datasets with both inflow and outflow, we reported the average results.
- Other Details: We introduced the time of the day as auxiliary data to improve prediction accuracy, following [56]. To ensure fairness of comparison, we only used the recent components of the ASTGCN [61] and ST-ResNet [14] model.

7.1.4 Performance and Insights

Table 6 represents the rank for all baselines. Tables 7, 8, and 9 present the experiment results for graph relation data, and Table 10 present the experiment results for grid relation data. From these results, we draw the following observations:

- General macro group prediction baseline methods, including Seq2Seq, AutoEncoder, and FNN, perform poorly on spatial-temporal data due to ignoring spatial correlation. This suggests that incorporating spatial correlation is crucial to improve prediction accuracy.
- Sequential Structure models, such as STResNet, ACFM, and DMVSTNet, which combine spatial CNNs and other modules, are only suitable for grid data as CNNs cannot handle graph data. Although these models are designed for grid-structured data, they perform worse than GNN-based models. One possible reason is that the original versions of such models require the input of information from more distant history, *i.e.,* data from the same period a week ago, which is removed in our experiments. Another reason could be that graph neural networks are capable of extracting spatial correlation from grid data if a suitable graph structure is built.
- Combining the results of all tables, we find that MTGNN is the best model, followed by GWNET. Both MTGNN and GWNET are sequence-structure models that combine GCNs and the Dilated Casual Convolution (also known as WaveNet) and use an adaptive graph structure learning module in the graph convolution part to overcome the shortage of fixed graph structures. Moreover, they perform direct prediction instead of a recurrent manner, solving the problem of cumulative errors. These models perform consistently well on different datasets. Thus, combining the structure of GCNs and the Dilated Casual Convolution is a promising research direction for future development in macro spatial-temporal prediction. Several recent works also follow this structure [50], [166].
- AGCRN is the better-performing models in order. AGCRN and DCRNN both use the coupled structure of GCNs and RNNs. AGCRN outperforms DCRNN by proposing a learnable graph structure and changing recurrent prediction to direct prediction, avoiding cumulative errors and speeding up training. In contrast, the model performs slightly worse when using sequential structures combining GCNs and RNNs, *i.e.,* TGCN and TGCLSTM. This indicates that the coupled structure is more appropriate for combining GCNs and RNNs than the sequential structures.
- As a sequential structural model using a combination of spatial attention and other components, GMAN ranks fourth overall. GMAN uses temporal and spatial attention mechanisms to learn spatial-temporal correlations in the data and performs well. However, the model has high complexity, large memory consumption, and slow computation speed. The other two attention models, *i.e.,* STTN and ASTGCN, perform worse than GMAN, probably due to differences in how attention is computed. GMAN uses a structure based entirely on attention mechanisms and introduces multi-headed attention and grouped attention, while STTN and ASTGCN have a simpler implementation of attention. The main limitation of attention-based models is their high time complexity, and subsequent studies can consider how to reduce the complexity of the model and achieve a balance between performance and efficiency.
- Last, models with spatial-temporal synchronized learning, *i.e.,* STG2Seq and STSGCN, perform slightly worse. This could be because they only consider short-range spatial-temporal correlations while ignoring long-range ones. MTGNN and GWNET, which use the multi-layer WaveNet, also aim to capture long-range temporal correlations. Some relatively recent works also verify the importance of long-range spatial dependence in spatial-temporal prediction [66], [167].

7.2 Benchmarks on Micro Individual Prediction Tasks

Here we use trajectory next-location predictions as a representative of micro individual prediction tasks for our experiments. The task of trajectory next-location prediction

is to predict the position of the next hop on the trajectory, given the current trajectory and the historical trajectories.

7.2.1 Datasets

We choose Foursquare-NYC [139] and Foursquare-TKY [139] dataset to test our reproduced models. The Foursquare dataset contains check-ins in NYC and Tokyo collected for about ten months (from Apr. 2012 to Feb. 2013). It contains 227,428 check-ins in New York City and 573,703 check-ins in Tokyo. Each check-in is associated with its time stamp, GPS coordinates, POI ID, and semantic meaning. These two datasets are the most commonly used open-source datasets in trajectory next-location prediction tasks, so it is more convincing to conduct comparative experiments on them.

7.2.2 Models

We conducted experiments on the following 11 models that can be divided into two categories as follows:

(1) Probability-based Models:

- FPMC [152]: combines Markov Chain with Matrix Factorization for sequence prediction.
- LSTM: uses a single-layer LSTM [68] to model sequences and uses an embedding layer to capture spatial-temporal features.
- ST-RNN [25]: introduces the spatial-temporal transition matrix into the hidden layer of RNN.
- HST-LSTM [90] introduces spatial-temporal transfer factors into LSTM and uses an encoder-decoder structure for prediction.
- SERM [89]: uses the pre-training WordVec to capture the semantic features in the trajectory.
- DeepMove [26]: extends GRU with the Attention mechanism to capture the correlation between current and historical trajectories.
- LSTPM [96]: captures long-term preferences with a non-local LSTM-based network and short-term preferences with a geo-dilated LSTM.

(2) Ranking-based Models:

- ATST-LSTM [91]: introduces the distance and time differences between the trajectory points into the LSTM and uses an Attention mechanism.
- GeoSAN [94]: uses a self-attention mechanism to realize the representation learning of POI.
- STAN [87]: uses a self-attention mechanism to capture spatial-temporal information to make predictions.
- CARA [95]: focuses on using the attention mechanism to extract contextual information between trajectories for prediction.

7.2.3 Experimental Settings

- Data Split and Preprocessing: We split all datasets into training, validation, and test sets using a 6:2:2 ratio. Trajectories with fewer than four check-ins and users with fewer than two trajectories are removed. This helps to eliminate the impact of inactive users on model performance. For each user, the first 60% of the trajectories are used for training, the middle 20% for validation, and the last 20% for testing.
- Prediction Window Settings: Except for GeoSAN and STAN, a time window size of 72 hours is used to cut the trajectory for all models. For GeoSAN and STAN, a length-based window size of 100 is used instead.
- Evaluation Metrics: We use four widely-used metrics to evaluate model performance: Recall@5, F1@5, MAP@5, and NDCG@5.
- Other Details: For the ranking-based model, we use a popularity-based negative sampler that selects 100 negative POIs based on their visit frequency. We chose this unified sampler to eliminate any unfairness in evaluating the experiments caused by using different samplers.

7.2.4 Performance and Insights

Table 11 presents the results of our experiments, where we use double horizontal lines to distinguish probability-based and ranking-based models. The state-of-the-art probability-based model is LSTPM [96], which demonstrates that the specially constructed LSTM in LSTPM, considering spatial locality and historical influences, can capture both long-term and short-term changes in trajectories more accurately, resulting in more precise predictions. This modeling paradigm, which introduces spatial-temporal patterns into the LSTM structure, can also be found in other reliable trajectory models such as ST-RNN [25] and HST-LSTM [90]. On the other hand, DeepMove [26] combines the Attention mechanism with the LSTM network and has achieved good results. Considering that the Attention mechanism is more widely used and successful in Ranking-based Models, the combination of LSTM and the Attention mechanism appears to be a feasible and promising research direction for modeling micro-individual trajectories in the future.

All of the ranking-based models use the Attention mechanism to rank positive and negative samples. However, since all models use a unified popularity-based sampler that differs from the one specifically constructed in the model's paper, the experimental results of the models on the two datasets show large fluctuations. Therefore, it is difficult to evaluate which paradigm is the most effective in ranking-based models.

8 Comparison with Existing Libraries

To the best of our knowledge, LibCity [6] is the first spatial-temporal prediction library that enables researchers to conduct comprehensive comparative experiments and develop new models. Recently, other researchers have proposed benchmarks in spatial-temporal prediction similar to LibCity. This section compares with others to demonstrate the advantages of LibCity.

DL-Traff [168] is an open-source project offering a traffic prediction benchmark using grid-based and graph-based models. DGCRN [169] summarizes previous work and produces a benchmark in traffic prediction. However, these two projects only accumulate the model codes from past research work without a modular design, which makes it inconvenient for users to use. Furthermore, they only focus on macro-level traffic prediction without contributing to micro-level individual prediction tasks. Microsoft FOST [14] (Forecasting Open Source Tool) is a general forecasting tool

TABLE 7
Experiments in Macro Group Prediction Tasks (Graph Relation Data Part 1)

Dataset	Horizon	Horizon 3			Horizon 6			Horizon 12			RANK
	Metric	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
METR-LA	Seq2Seq	4.6	10.81	7.75	4.74	11.51	8.2	5.07	12.89	9.04	14
	AutoEncoder	4.03	11.99	7.75	4.10	12.21	7.85	4.26	12.91	8.29	13
	FNN	3.84	10.41	7.48	4.56	12.31	8.30	5.21	16.57	9.82	15
	STGCN	2.76	7.02	5.42	3.34	8.97	6.86	4.31	11.75	8.79	8
	GWNET	2.69	6.90	5.18	3.07	8.37	6.22	3.53	10.01	7.35	2
	MTGNN	2.67	6.89	5.14	3.06	8.19	6.16	3.52	10.03	7.28	1
	TGCN	3.06	8.18	5.75	3.55	10.05	6.88	4.17	12.36	8.24	12
	TGCLSTM	3.08	8.77	6.40	4.09	8.83	6.82	4.18	12.08	8.28	11
	ASTGCN	3.02	8.21	5.76	3.52	10.00	6.86	4.12	12.07	8.14	10
	GMAN	2.81	7.43	5.51	3.15	8.67	6.50	3.59	10.33	7.51	3
	STTN	2.91	7.82	5.67	3.33	9.12	6.67	3.76	10.61	7.57	6
	DCRNN	2.72	7.03	5.38	3.15	8.67	6.44	3.67	10.81	7.72	4
	AGCRN	2.88	7.71	5.55	3.27	9.07	6.56	3.66	10.49	7.51	5
	STG2Seq	3.01	8.01	5.88	3.45	9.18	6.66	3.82	11.02	7.72	9
	STSGCN	2.93	7.88	5.78	3.34	9.09	6.68	3.80	10.93	7.65	7
PEMS-BAY	Seq2Seq	2.56	6.04	5.18	2.57	6.03	5.24	2.78	6.45	5.72	15
	AutoEncoder	2.50	6.05	5.16	2.52	6.10	5.19	2.64	6.45	5.44	13
	FNN	1.51	5.14	3.33	2.07	5.56	4.81	2.87	7.93	6.50	13
	STGCN	1.35	2.84	2.90	1.80	4.12	4.19	2.37	5.79	5.61	8
	GWNET	1.32	2.77	2.73	1.63	3.67	3.67	1.95	4.51	4.55	1
	MTGNN	1.34	2.83	2.79	1.67	3.74	3.74	1.99	4.65	4.62	2
	TGCN	1.50	3.17	3.03	1.86	4.24	3.99	2.27	5.43	4.87	10
	TGCLSTM	1.39	3.21	3.17	1.92	4.22	4.12	2.33	5.45	4.96	11
	ASTGCN	1.46	3.05	3.06	1.94	4.37	4.32	2.52	5.96	5.58	12
	GMAN	1.38	2.93	2.95	1.69	3.80	3.87	1.94	4.54	4.49	3
	STTN	1.45	3.16	3.04	1.76	4.05	3.97	2.06	4.88	4.71	6
	DCRNN	1.36	2.93	2.89	1.74	3.88	3.97	2.06	4.89	4.73	5
	AGCRN	1.37	2.95	2.89	1.70	3.84	3.85	1.99	4.61	4.60	4
	STG2Seq	1.39	3.01	3.10	1.88	4.14	4.04	2.12	4.98	4.73	9
	STSGCN	1.38	2.97	3.12	1.89	3.92	4.07	2.09	4.91	4.70	7
PEMSD7(M)	Seq2Seq	3.51	9.09	6.62	3.56	9.27	6.80	4.34	10.88	7.95	13
	AutoEncoder	3.74	9.95	7.08	3.80	10.17	7.18	4.17	10.74	7.96	15
	FNN	3.37	6.52	6.50	3.51	7.78	6.52	4.67	12.16	8.93	14
	STGCN	2.20	5.16	4.05	2.94	7.24	5.66	3.79	9.46	7.22	5
	GWNET	2.12	5.05	4.03	2.73	6.89	5.46	3.20	8.26	6.44	2
	MTGNN	2.12	4.99	4.01	2.71	6.67	5.42	3.19	8.02	6.47	1
	TGCN	2.66	5.35	4.20	2.93	7.50	6.16	4.21	10.58	7.77	10
	TGCLSTM	2.67	5.39	4.36	2.95	7.53	6.23	4.33	10.55	7.82	12
	ASTGCN	2.31	5.47	4.24	3.11	7.76	5.99	4.04	10.53	7.75	11
	GMAN	2.34	5.43	4.19	2.85	6.86	5.32	3.34	8.26	6.26	4
	STTN	2.36	5.31	4.22	2.93	7.47	6.11	3.90	10.24	7.36	8
	DCRNN	2.20	5.15	4.20	2.98	7.45	5.91	3.94	10.49	7.78	7
	AGCRN	2.16	5.24	4.08	2.74	6.93	5.50	3.25	8.41	6.62	3
	STG2Seq	2.23	5.18	4.24	3.02	7.52	6.12	4.02	10.35	7.82	9
	STSGCN	2.21	5.16	4.18	2.93	7.44	6.02	3.96	10.36	7.83	6

that aims to provide an easy-to-use tool for spatial-temporal forecasting, but its supported models and applications are limited.

Table 12 illustrates that LibCity outperforms the other compared tools in terms of the number of models and datasets it can handle. Furthermore, LibCity designs a unified storage format for spatial-temporal data, which is a great help to promote the standardization of the field. The modular design of LibCity allows for scalability and enables developers to easily create new models using its pipeline. The high number of Stars and Forks indicates that LibCity is popular among the open-source community. The number of open issues and total issues also suggests that the developers of LibCity are active in the community, which helps to address user inquiries and promote further development in the field.

In addition, it is worth noting that numerous similar experimental libraries are available in other research fields. For instance, RecBole [10] is a recommendation algorithm framework that reproduces a vast range of recommendation models and provides various evaluation strategies and data

14. https://github.com/microsoft/FOST

TABLE 8
Experiments in Macro Group Prediction Tasks (Graph Relation Data Part 2)

Dataset	Horizon	Horizon 3			Horizon 6			Horizon 12			RANK
	Metric	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
PEMSD4-Speed	Seq2Seq	2.23	5.04	4.85	2.24	5.04	4.89	2.33	5.24	5.06	13
	AutoEncoder	2.32	5.41	5.11	2.33	5.43	5.14	2.38	5.56	5.26	15
	FNN	1.85	4.83	4.11	2.19	4.94	4.72	2.51	6.67	6.71	14
	STGCN	1.40	2.71	2.87	1.78	3.55	3.91	2.26	4.91	5.00	7
	GWNET	1.32	2.57	2.79	1.62	3.37	3.67	1.91	4.15	4.45	1
	MTGNN	1.33	2.62	2.84	1.64	3.44	3.77	1.93	4.25	4.55	3
	TGCN	1.62	2.92	2.97	1.92	3.86	3.92	2.29	4.74	4.69	11
	TGCLSTM	1.77	3.06	3.05	1.94	3.84	4.12	2.30	4.87	4.93	12
	ASTGCN	1.42	2.80	2.97	1.80	3.90	4.03	2.25	4.92	5.15	10
	GMAN	1.38	3.03	2.94	1.74	3.68	3.69	2.03	4.37	4.44	5
	STTN	1.41	2.88	2.98	1.66	3.55	3.79	1.90	4.21	4.47	4
	DCRNN	1.34	2.65	2.88	1.69	3.59	3.89	2.05	4.64	4.82	6
	AGCRN	1.35	2.69	2.83	1.64	3.46	3.71	1.92	4.22	4.46	2
	STG2Seq	1.48	3.02	3.13	1.91	3.72	3.92	2.29	4.32	4.53	9
	STSGCN	1.46	2.94	3.14	1.86	3.64	3.87	2.26	4.24	4.46	8
PEMSD8-Speed	Seq2Seq	1.89	4.58	4.57	1.91	4.63	4.64	1.96	4.75	4.77	13
	AutoEncoder	2.15	5.19	5.18	2.17	5.21	5.22	2.25	5.36	5.33	15
	FNN	1.67	4.30	4.64	1.95	4.59	4.78	2.11	6.23	6.64	14
	STGCN	1.21	2.57	2.61	1.56	3.33	3.43	1.74	4.35	4.28	6
	GWNET	1.10	2.20	2.53	1.36	2.97	3.44	1.57	3.65	4.13	2
	MTGNN	1.11	2.20	2.48	1.36	2.94	3.30	1.61	3.84	4.13	1
	TGCN	1.58	3.18	2.87	1.82	3.74	3.55	2.16	4.65	4.36	11
	TGCLSTM	1.62	3.19	2.91	1.93	3.80	3.61	2.14	4.72	4.35	12
	ASTGCN	1.20	2.40	2.58	1.54	3.46	3.63	1.88	4.43	4.55	8
	GMAN	1.26	2.66	2.69	1.46	3.22	3.33	1.68	3.84	3.92	5
	STTN	1.30	3.00	3.17	1.50	3.56	3.83	1.66	4.03	4.34	7
	DCRNN	1.14	2.30	2.55	1.43	3.14	3.47	1.73	4.08	4.33	4
	AGCRN	1.14	2.39	2.57	1.39	3.06	3.35	1.62	3.73	4.08	3
	STG2Seq	1.37	3.15	3.24	1.59	3.55	3.91	1.83	4.17	4.48	10
	STSGCN	1.35	3.14	3.26	1.55	3.54	3.85	1.78	4.11	4.39	9
PEMSD4-Flow	Seq2Seq	22.81	16.21	36.67	22.65	16.11	36.46	23.32	16.81	37.19	13
	AutoEncoder	24.28	17.70	38.13	24.27	17.71	38.09	25.27	18.42	39.37	15
	FNN	21.68	15.81	35.57	23.75	16.91	37.10	29.93	20.71	46.01	14
	STGCN	19.01	12.90	29.90	20.77	14.03	32.41	23.39	15.80	36.20	7
	GWNET	17.77	12.51	28.67	18.66	13.20	30.18	20.01	14.51	32.25	1
	MTGNN	18.06	13.21	29.14	18.82	14.20	30.51	20.00	15.31	32.34	3
	TGCN	20.87	14.32	32.39	20.87	15.08	33.90	21.49	15.48	34.34	11
	TGCLSTM	21.10	14.49	32.75	21.28	15.16	34.07	21.34	15.47	34.78	12
	ASTGCN	19.51	13.05	30.76	21.20	14.17	33.49	21.57	16.06	34.61	9
	GMAN	18.46	13.80	29.34	19.30	14.32	30.71	20.77	15.70	32.64	4
	STTN	19.06	13.74	30.21	19.88	14.61	31.41	21.33	16.01	33.43	6
	DCRNN	18.72	12.81	29.72	19.97	13.81	31.66	22.02	15.40	34.72	5
	AGCRN	18.20	12.62	29.34	18.94	13.22	30.72	20.02	13.92	32.34	2
	STG2Seq	19.43	13.93	30.63	20.05	14.80	31.35	21.76	16.18	33.39	9
	STSGCN	19.31	13.86	30.22	19.93	14.77	31.45	21.51	16.14	33.34	7

preprocessing operations, making it easy to conduct experiments. Meanwhile, MMDetection [9] adopts a modular design, enabling researchers to efficiently develop new models based on it for object detection tasks. FastReID [170] continuously reproduces state-of-the-art models and releases corresponding pre-trained models for both research and industrial purposes.

LibCity combines the strengths of the libraries as mentioned above, such as various baseline models, diverse evaluation strategies, and modular design. As a result, it not only facilitates researchers to conduct experiments and develop new models but also promotes standardization within the spatial-temporal prediction field.

9 Conclusion

In this work, we present a comprehensive review of urban spatial-temporal prediction and proposes a unified storage format for spatial-temporal data, called atomic files. Building on this, we introduce LibCity, a unified and comprehensive open-source library for urban spatial-temporal prediction that includes 55 spatial-temporal datasets and

TABLE 9
Experiments in Macro Group Prediction Tasks (Graph Relation Data Part 3)

Dataset	Horizon	Horizon 3			Horizon 6			Horizon 12			RANK
	Metric	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
PEMSD8-Flow	Seq2Seq	20.16	12.3	32.88	20.25	12.42	33.05	20.88	14.61	33.87	14
	AutoEncoder	21.24	13.60	34.71	21.41	13.71	34.88	22.34	14.42	35.98	15
	FNN	18.39	11.44	27.50	19.92	12.11	31.94	24.24	16.21	37.57	13
	STGCN	14.83	9.93	22.48	15.69	10.43	24.47	17.23	11.61	27.44	5
	GWNET	13.51	8.61	21.58	14.37	9.40	23.21	15.70	10.41	25.54	1
	MTGNN	14.21	9.52	22.31	15.10	10.41	23.97	16.53	10.82	26.18	3
	TGCN	15.63	10.48	24.48	16.10	11.42	24.72	17.38	14.12	27.97	10
	TGCLSTM	15.78	10.50	24.45	16.29	11.75	24.83	17.46	14.32	27.77	12
	ASTGCN	15.64	10.73	23.06	16.23	11.49	24.50	17.87	14.30	27.85	11
	GMAN	14.72	10.20	22.81	15.61	10.81	24.40	17.56	12.31	27.13	6
	STTN	15.56	10.29	22.56	15.94	10.87	24.56	16.98	12.87	27.75	7
	DCRNN	14.35	9.32	22.39	15.43	10.01	24.40	17.15	11.05	27.23	4
	AGCRN	13.98	9.91	22.11	14.86	9.60	24.09	16.19	10.51	26.32	2
	STG2Seq	15.68	10.34	22.85	16.18	10.92	24.74	17.19	12.94	27.93	9
	STSGCN	15.63	10.25	22.71	16.11	10.85	24.65	17.04	12.92	27.86	8
PEMSD4-Occupy	Seq2Seq	0.92	20.57	2.38	0.92	20.55	2.38	0.98	21.39	2.54	13
	AutoEncoder	0.97	20.69	2.47	0.98	20.98	2.48	1.04	22.18	2.62	15
	FNN	0.87	18.94	2.18	0.92	20.92	2.31	1.20	27.09	2.86	14
	STGCN	0.68	15.18	1.80	0.78	16.65	2.13	0.86	18.48	2.47	5
	GWNET	0.67	15.16	1.74	0.74	16.32	2.02	0.82	18.02	2.24	2
	MTGNN	0.68	15.28	1.73	0.73	16.20	1.97	0.81	17.48	2.20	1
	TGCN	0.80	15.99	1.85	0.84	16.91	2.20	0.94	20.70	2.49	11
	TGCLSTM	0.83	16.06	1.89	0.87	16.97	2.24	0.97	21.05	2.52	12
	ASTGCN	0.73	16.02	1.83	0.84	16.82	2.21	0.97	20.63	2.52	10
	GMAN	0.70	15.63	1.78	0.77	16.63	2.05	0.84	18.31	2.26	4
	STTN	0.76	15.65	1.82	0.80	16.72	2.11	0.85	19.35	2.49	7
	DCRNN	0.69	15.54	1.79	0.77	16.79	2.09	0.87	18.47	2.36	6
	AGCRN	0.68	15.49	1.75	0.74	16.58	2.02	0.82	18.03	2.23	3
	STG2Seq	0.79	15.96	1.82	0.83	16.87	2.21	0.91	20.31	2.51	9
	STSGCN	0.77	15.82	1.83	0.82	16.76	2.19	0.87	19.69	2.52	8
PEMSD8-Occupy	Seq2Seq	0.89	14.34	2.38	0.89	14.39	2.40	0.92	14.86	2.44	14
	AutoEncoder	0.99	15.56	2.57	0.99	15.77	2.58	1.02	16.51	2.60	15
	FNN	0.78	12.81	2.14	0.82	14.17	2.13	1.07	19.29	2.47	13
	STGCN	0.61	10.62	1.64	0.71	11.97	1.94	0.85	12.29	2.24	6
	GWNET	0.59	9.87	1.63	0.66	10.67	1.91	0.74	11.96	2.15	1
	MTGNN	0.60	10.09	1.59	0.67	10.90	1.91	0.75	12.05	2.17	2
	TGCN	0.71	11.54	1.75	0.77	12.16	1.99	0.87	12.80	2.29	11
	TGCLSTM	0.73	11.65	1.80	0.79	12.29	2.03	0.89	12.94	2.32	12
	ASTGCN	0.66	11.20	1.66	0.76	12.28	1.97	0.88	12.47	2.24	9
	GMAN	0.63	11.08	1.67	0.69	11.40	1.89	0.76	12.47	2.12	4
	STTN	0.63	11.16	1.72	0.70	11.80	1.93	0.81	12.64	2.25	7
	DCRNN	0.62	10.29	1.65	0.69	11.30	1.92	0.80	12.67	2.19	4
	AGCRN	0.62	10.90	1.66	0.68	11.36	1.88	0.76	12.37	2.08	3
	STG2Seq	0.69	11.38	1.74	0.73	12.06	1.97	0.85	12.74	2.28	10
	STSGCN	0.67	11.27	1.73	0.72	11.93	1.94	0.81	12.68	2.26	8

65 spatial-temporal prediction models covering 9 mainstream sub-tasks of urban spatial-temporal prediction. By conducting extensive experiments using LibCity, we establish a comprehensive model performance leaderboard that identifies promising research directions for spatial-temporal prediction.

To the best of our knowledge, LibCity is the first open-source library for urban spatial-temporal prediction, providing a valuable tool for exploring and developing spatial-temporal prediction models. We will continuously expand LibCity to contribute to the spatial-temporal prediction field in the future. For example, we can cover more spatial-temporal prediction tasks, such as climate prediction, air quality prediction, theft prediction, etc.

REFERENCES

[1] X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi, and B. Yin, "Deep learning on traffic prediction: Methods, analysis and future directions," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[2] K. Gu, J. Hu, and W. Jia, "Adaptive area-based traffic congestion control and management scheme based on fog computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 1359–1373, 2023.

TABLE 10
Experiments in Macro Group Prediction Tasks (Grid Relation Data)

Dataset	NYCTaxi150103			NYCTaxi160102			NYCTaxi140103			TaxiBJ2014			TaxiBJ2015			
Metric	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	RANK
Seq2Seq	12.95	20.21	21.23	12.89	20.35	21.23	8.92	21.90	11.47	16.94	18.93	21.93	18.29	21.16	27.88	18
AutoEncoder	12.56	20.86	21.90	12.83	20.11	22.37	8.97	20.18	11.65	15.00	17.31	21.05	18.53	22.57	22.84	17
FNN	14.12	22.32	21.68	13.65	22.22	19.84	8.82	20.16	11.46	14.43	17.52	21.74	15.99	18.69	21.62	16
STResNet	12.12	20.18	20.71	11.96	18.81	16.59	9.09	19.96	11.84	11.82	15.64	18.83	12.85	16.87	18.93	15
ACFM	12.24	20.06	20.53	11.40	18.78	16.31	8.77	19.70	11.63	11.74	15.83	18.71	12.83	16.33	18.90	14
DMVSTNet	11.99	19.97	20.47	11.53	18.61	16.23	8.75	19.66	11.40	11.71	15.64	18.67	12.67	15.24	18.75	13
STGCN	10.84	18.48	18.64	10.68	18.58	14.95	8.44	18.71	9.99	11.47	14.46	18.63	11.62	14.58	17.34	6
GWNET	10.24	17.37	16.71	10.48	17.98	15.28	8.07	18.56	10.39	11.43	13.98	18.34	11.22	14.71	18.38	4
MTGNN	10.42	17.19	15.80	10.00	16.80	14.50	8.02	18.33	9.86	11.48	14.05	17.38	11.03	14.37	17.05	1
TGCN	11.97	18.93	20.27	10.11	18.16	14.86	8.74	18.91	11.37	11.62	15.59	18.18	11.87	15.11	18.72	12
TGCLSTM	11.13	19.33	19.47	9.81	18.15	15.27	8.73	18.67	11.30	11.69	15.57	17.89	11.99	15.21	18.51	10
ASTGCN	11.97	19.99	20.32	10.48	18.19	14.83	8.50	19.51	11.33	11.44	14.35	18.33	12.27	15.27	17.63	11
GMAN	9.83	17.62	15.53	9.61	17.60	14.12	7.67	18.04	9.76	11.48	14.26	18.05	11.23	15.17	17.58	2
STTN	10.55	20.05	17.20	9.86	18.73	14.50	8.70	18.99	10.87	11.20	15.15	18.62	11.36	15.02	18.04	7
DCRNN	10.00	17.81	14.86	9.82	17.53	14.00	8.52	19.58	10.46	11.68	14.70	17.44	11.36	15.11	17.56	5
AGCRN	10.02	17.52	17.92	9.79	17.30	14.39	7.76	18.19	10.39	11.40	14.14	18.25	11.12	14.48	18.06	3
STG2Seq	11.21	18.82	18.27	11.27	17.63	15.41	8.43	19.85	10.53	11.63	14.36	18.52	11.33	15.21	18.10	9
STSGCN	10.93	18.95	16.94	10.28	17.74	15.24	8.08	19.55	10.98	11.69	14.38	18.53	11.56	15.11	18.45	8
Dataset	NYCBike160708			NYCBike140409			NYCBike200709			BikeDC			BikeCHI			
Metric	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	RANK
Seq2Seq	5.61	26.14	7.35	5.61	25.83	7.81	8.57	25.52	14.34	10.88	34.97	15.09	5.80	27.92	8.53	16
AutoEncoder	5.81	26.79	7.55	5.63	25.72	7.89	8.59	25.54	14.58	10.94	32.57	14.83	5.88	28.72	8.68	17
FNN	5.80	27.12	7.90	6.15	25.59	7.77	8.62	25.43	14.72	10.87	32.77	14.99	6.12	29.38	8.78	18
STResNet	5.53	25.85	7.05	5.86	26.63	7.97	8.86	25.40	14.18	10.77	30.38	14.89	5.62	27.19	7.96	15
ACFM	5.64	25.89	6.87	5.68	26.30	7.85	8.75	25.38	14.27	10.73	30.25	14.83	5.54	26.66	7.90	14
DMVSTNet	5.32	25.80	6.76	5.63	26.29	7.75	8.70	25.32	14.08	10.67	30.15	14.70	5.53	26.41	7.71	13
STGCN	4.64	23.21	6.53	5.31	24.96	6.91	7.52	22.30	11.31	10.10	27.13	13.66	4.67	24.84	6.42	6
GWNET	4.55	23.22	6.48	5.19	24.19	6.85	7.47	22.38	11.11	10.14	26.30	12.88	4.72	24.61	6.24	4
MTGNN	4.49	22.58	5.87	4.94	23.57	6.41	7.10	21.97	10.36	9.96	26.89	12.68	4.50	24.60	5.69	1
TGCN	4.82	25.81	6.51	5.46	24.43	7.07	8.55	25.34	13.74	10.06	29.68	14.63	5.51	26.40	6.77	11
TGCLSTM	4.87	25.62	6.69	5.53	24.85	7.47	8.53	25.36	13.97	10.30	29.10	14.60	5.51	26.38	6.50	12
ASTGCN	5.22	24.53	6.68	5.57	24.13	7.44	8.27	25.21	13.82	10.16	29.04	14.34	5.42	26.41	5.87	10
GMAN	4.53	22.90	5.95	5.03	24.44	6.85	7.43	22.24	11.40	9.87	26.11	12.11	4.75	24.60	6.25	2
STTN	4.65	24.22	6.37	5.14	25.24	6.80	7.50	23.49	11.72	9.97	27.79	12.22	4.81	25.75	6.60	7
DCRNN	4.50	23.13	6.38	5.12	24.47	6.93	7.44	22.43	11.42	9.95	27.11	13.26	4.73	25.62	6.21	5
AGCRN	4.52	23.02	5.97	5.13	24.43	6.65	7.54	22.38	11.45	9.96	27.57	12.62	4.78	25.61	6.11	3
STG2Seq	5.08	24.26	6.74	5.41	24.46	7.09	8.04	25.05	11.93	10.11	28.09	13.77	4.80	25.86	6.58	9
STSGCN	4.63	23.66	6.64	5.40	24.67	7.18	7.62	23.16	11.81	9.97	27.61	12.99	4.81	26.21	6.44	8

TABLE 11
Experiments in Trajectory Next-Location Prediction

Model\Dataset	Foursquare-TKY				Foursuqare-NYC			
	Recall@5	F1-score@5	MAP@5	NDCG@5	Recall@5	F1-score@5	MAP@5	NDCG@5
FPMC	0.0823	0.0274	0.0511	0.0588	0.2156	0.0719	0.1381	0.1574
LSTM	0.3141	0.1047	0.2155	0.2400	0.2449	0.0816	0.1610	0.1819
ST-RNN	0.0724	0.0241	0.0334	0.0431	0.0331	0.0110	0.0196	0.0230
HSTLSTM	0.2853	0.0951	0.1935	0.2164	0.2391	0.0797	0.1559	0.1766
SERM	0.3147	0.1049	0.2024	0.2024	0.2740	0.0913	0.1807	0.2040
DeepMove	0.3350	0.1117	0.2215	0.2498	0.3264	0.1088	**0.2141**	**0.2421**
LSTPM	**0.3624**	**0.1208**	**0.2411**	**0.2713**	**0.3307**	**0.1102**	0.2101	0.2401
STAN	0.1835	0.0612	0.1195	0.1354	0.1966	0.0655	0.1158	0.1358
CARA	0.3688	0.1229	0.2577	0.2854	0.1657	0.0552	0.0852	0.1051
ATST-LSTM	0.4953	0.1651	**0.3530**	**0.3886**	**0.4427**	**0.1476**	**0.2569**	**0.3029**
GeoSAN	**0.5157**	**0.1719**	0.3232	0.3710	0.3218	0.1073	0.1987	0.2293

TABLE 12
Comparison with existing libraries by 2023/4/22

Framework\Metric	Modularization	#Fork	#Star	#Models	#Datasets	#Issues
DL-Traff	Low	34	179	18	7	2/3
DGCRN	Low	64	173	12	3	9/13
FOST	Middle	43	208	6	0	4/13
LibCity	**High**	**111**	**485**	**65**	**55**	**0/73**

[3] B. Li, T. Dai, W. Chen, X. Song, Y. Zang, Z. Huang, Q. Lin, and K. Cai, "T-PORP: A trusted parallel route planning model on dynamic road networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 1238–1250, 2023.

[4] M. Zhou, J. Jin, W. Zhang, Z. T. Qin, Y. Jiao, C. Wang, G. Wu, Y. Yu, and J. Ye, "Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching," in *CIKM*. ACM, 2019, pp. 2645–2653.

[5] H. Zang, D. Han, X. Li, Z. Wan, and M. Wang, "CHA: categorical hierarchy-based attention for next POI recommendation," *ACM Trans. Inf. Syst.*, vol. 40, no. 1, pp. 7:1–7:22, 2022.

[6] J. Wang, J. Jiang, W. Jiang, C. Li, and W. X. Zhao, "Libcity: An open library for traffic prediction," in *SIGSPATIAL/GIS*. ACM, 2021, pp. 145–148.

[7] D. A. Tedjopurnomo, Z. Bao, B. Zheng, F. M. Choudhury, and A. K. Qin, "A survey on modern deep neural network for traffic prediction: Trends, methods and challenges," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 4, pp. 1544–1561, 2022.

[8] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*. IEEE Computer Society, 2009, pp. 248–255.

[9] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "Mmdetection: Open mmlab detection toolbox and benchmark," *CoRR*, vol. abs/1906.07155, 2019.

[10] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian, Y. Min, Z. Feng, X. Fan, X. Chen, P. Wang, W. Ji, Y. Li, X. Wang, and J. Wen, "Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms," in *CIKM*. ACM, 2021, pp. 4653–4664.

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019, pp. 8024–8035.

[12] J. Ye, J. Zhao, K. Ye, and C. Xu, "How to build a graph-based deep learning architecture in traffic domain: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 5, pp. 3904–3924, 2022.

[13] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "Dnn-based prediction model for spatio-temporal data," in *SIGSPATIAL/GIS*. ACM, 2016, pp. 92:1–92:4.

[14] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[15] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations (ICLR '18)*, 2018.

[16] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[17] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *AAAI*. AAAI Press, 2018, pp. 2588–2595.

[18] S. Guo, C. Chen, J. Wang, Y. Liu, K. Xu, Z. Yu, D. Zhang, and D. M. Chiu, "Rod-revenue: Seeking strategies analysis and revenue prediction in ride-on-demand service using multi-source urban data," *IEEE Trans. Mob. Comput.*, vol. 19, no. 9, pp. 2202–2220, 2020.

[19] Z. Duan, K. Zhang, Z. Chen, Z. Liu, L. Tang, Y. Yang, and Y. Ni, "Prediction of city-scale dynamic taxi origin-destination flows using a hybrid deep neural network combined with travel time," *IEEE Access*, vol. 7, pp. 127 816–127 832, 2019.

[20] L. Han, X. Ma, L. Sun, B. Du, Y. Fu, W. Lv, and H. Xiong, "Continuous-time and multi-level graph representation learning for origin-destination demand prediction," in *KDD*. ACM, 2022, pp. 516–524.

[21] Q. Chen, X. Song, H. Yamada, and R. Shibasaki, "Learning deep representation from big and heterogeneous data for traffic accident inference," in *AAAI*. AAAI Press, 2016, pp. 338–344.

[22] Z. Zhou, Y. Wang, X. Xie, L. Chen, and H. Liu, "Riskoracle: A minute-level citywide traffic accident forecasting framework," in *AAAI*. AAAI Press, 2020, pp. 1258–1265.

[23] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, pp. 29:1–29:41, 2015.

[24] S. Wang, Z. Bao, J. S. Culpepper, and G. Cong, "A survey on trajectory data management, analytics, and learning," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 39:1–39:36, 2022.

[25] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, p. 194–200.

[26] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, "Deepmove: Predicting human mobility with attentional recurrent networks," in *WWW*. ACM, 2018, pp. 1459–1468.

[27] R. Sevlian and R. Rajagopal, "Travel time estimation using floating car data," *CoRR*, vol. abs/1012.4249, 2010.

[28] B. Pan, U. Demiryurek, and C. Shahabi, "Utilizing real-world transportation data for accurate traffic prediction," in *ICDM*. IEEE Computer Society, 2012, pp. 595–604.

[29] H. Alt, A. Efrat, G. Rote, and C. Wenk, "Matching planar maps," in *SODA*. ACM/SIAM, 2003, pp. 589–598.

[30] T. Yaqub, M. J. Tordon, and J. Katupitiya, "Line segment based scan matching for concurrent mapping and localization of a mobile robot," in *ICARCV*. IEEE, 2006, pp. 1–6.

[31] T. S. Jepsen, C. S. Jensen, and T. D. Nielsen, "Graph convolutional networks for road networks," in *SIGSPATIAL/GIS*. ACM, 2019, pp. 460–463.

[32] M. Wang, W. Lee, T. Fu, and G. Yu, "Learning embeddings of intersections on road networks," in *SIGSPATIAL/GIS*. ACM, 2019, pp. 309–318.

[33] J. D. Hamilton, *Time series analysis*. Princeton university press, 1994.

[34] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results," *Journal of transportation engineering*, vol. 129, no. 6, pp. 664–672, 2003.

[35] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *NIPS*. MIT Press, 1996, pp. 155–161.

[36] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, "Traffic speed prediction and congestion source exploration: A deep learning method," in *ICDM*. IEEE Computer Society, 2016, pp. 499–508.

[37] L. Liu, R. Zhang, J. Peng, G. Li, B. Du, and L. Lin, "Attentive crowd flow machines," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 1553–1561.

[38] H. Lin, R. Bai, W. Jia, X. Yang, and Y. You, "Preserving dynamic attention for long-term spatial-temporal prediction," in *KDD*. ACM, 2020, pp. 36–46.

[39] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-gcn: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2019.

[40] J. Ji, J. Wang, Z. Jiang, J. Ma, and H. Zhang, "Interpretable spatiotemporal deep learning model for traffic flow prediction based on potential energy fields," in *ICDM*. IEEE, 2020, pp. 1076–1081.

[41] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 914–921.

[42] W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao, and X. Feng, "Multi-range attentive bicomponent graph convolutional network for traffic forecasting," in *AAAI*. AAAI Press, 2020, pp. 3529–3536.

[43] Q. Zhang, J. Chang, G. Meng, S. Xiang, and C. Pan, "Spatio-temporal graph structure learning for traffic forecasting," in *AAAI*. AAAI Press, 2020, pp. 1177–1185.

[44] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, and S. He, "Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting," in *AAAI*. AAAI Press, 2019, pp. 890–897.

[45] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *KDD*, 2020, pp. 753–763.

[46] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," *NIPS*, vol. 33, 2020.

[47] X. Zhang, C. Huang, Y. Xu, L. Xia, P. Dai, L. Bo, J. Zhang, and Y. Zheng, "Traffic flow forecasting with spatial-temporal graph diffusion network," in *AAAI*. AAAI Press, 2021, pp. 15 008–15 015.

[48] J. Ye, L. Sun, B. Du, Y. Fu, and H. Xiong, "Coupled layer-wise graph convolution for transportation demand prediction," in *AAAI*. AAAI Press, 2021, pp. 4617–4625.

[49] B. N. Oreshkin, A. Amini, L. Coyle, and M. Coates, "FC-GAGA: fully connected gated graph architecture for spatio-temporal traffic forecasting," in *AAAI*. AAAI Press, 2021, pp. 9233–9241.

[50] L. Han, B. Du, L. Sun, Y. Fu, Y. Lv, and H. Xiong, "Dynamic and multi-faceted spatio-temporal deep learning for traffic speed forecasting," in *KDD*. ACM, 2021, pp. 547–555.

[51] J. Choi, H. Choi, J. Hwang, and N. Park, "Graph neural controlled differential equations for traffic forecasting," in *AAAI*. AAAI Press, 2022, pp. 6367–6374.

[52] J. Ji, J. Wang, Z. Jiang, J. Jiang, and H. Zhang, "STDEN: towards physics-guided neural networks for traffic flow prediction," in *AAAI*. AAAI Press, 2022, pp. 4048–4056.

[53] D. Liu, J. Wang, S. Shang, and P. Han, "MSDR: multi-step dependency relation networks for spatial temporal forecasting," in *KDD*. ACM, 2022, pp. 1042–1050.

[54] L. Han, X. Ma, L. Sun, B. Du, Y. Fu, W. Lv, and H. Xiong, "Continuous-time and multi-level graph representation learning for origin-destination demand prediction," in *KDD*. ACM, 2022, pp. 516–524.

[55] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016, pp. 3837–3845.

[56] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *IJCAI*. International Joint Conferences on Artificial Intelligence Organization, 2019.

[57] S. Guo, Y. Lin, H. Wan, X. Li, and G. Cong, "Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 11, pp. 5415–5428, 2022.

[58] Z. Fang, Q. Long, G. Song, and K. Xie, "Spatial-temporal graph ODE networks for traffic flow forecasting," in *KDD*. ACM, 2021, pp. 364–373.

[59] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D. Yeung, "Gaan: Gated attention networks for learning on large and spatiotemporal graphs," in *UAI*. AUAI Press, 2018, pp. 339–349.

[60] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A graph multi-attention network for traffic prediction," in *AAAI*. AAAI Press, 2020, pp. 1234–1241.

[61] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 922–929.

[62] B. Lu, X. Gan, H. Jin, L. Fu, and H. Zhang, "Spatiotemporal adaptive gated graph convolution network for urban traffic flow forecasting," in *CIKM*. ACM, 2020, pp. 1025–1034.

[63] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G. Qi, and H. Xiong, "Spatial-temporal transformer networks for traffic flow forecasting," *CoRR*, vol. abs/2001.02908, 2020.

[64] H. Yan, X. Ma, and Z. Pu, "Learning dynamic and hierarchical traffic spatiotemporal features with transformer," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 22 386–22 399, 2022.

[65] X. Ye, S. Fang, F. Sun, C. Zhang, and S. Xiang, "Meta graph transformer: A novel framework for spatial-temporal traffic prediction," *Neurocomputing*, vol. 491, pp. 544–563, 2022.

[66] J. Jiang, C. Han, W. X. Zhao, and J. Wang, "Pdformer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction," in *AAAI*. AAAI Press, 2023.

[67] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4883–4894, 2019.

[68] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[69] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[70] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014, pp. 3104–3112.

[71] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *SSW*. ISCA, 2016, p. 125.

[72] K. Guo, Y. Hu, Y. Sun, S. Qian, J. Gao, and B. Yin, "Hierarchical graph convolution network for traffic forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 1, 2021, pp. 151–159.

[73] K. Guo, Y. Hu, Z. S. Qian, Y. Sun, J. Gao, and B. Yin, "Dynamic graph convolution network for traffic forecasting based on latent network of laplace matrix estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1009–1018, 2022.

[74] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Q. Z. Sheng, "Stg2seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting," in *IJCAI*. ijcai.org, 2019, pp. 1981–1987.

[75] H. Qiu, Q. Zheng, M. Msahli, G. Memmi, M. Qiu, and J. Lu, "Topological graph convolutional network-based urban traffic flow and density prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4560–4569, 2021.

[76] M. Li and Z. Zhu, "Spatial-temporal fusion graph neural networks for traffic flow forecasting," in *AAAI*. AAAI Press, 2021, pp. 4189–4196.

[77] T.-Y. Fu and W.-C. Lee, "Trembr: Exploring road networks for trajectory representation learning," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 1, pp. 1–25, 2020.

[78] J. Jiang, D. Pan, H. Ren, X. Jiang, C. Li, and J. Wang, "Self-supervised trajectory representation learning with temporal regularities and travel semantics," in *2023 IEEE 39th international conference on data engineering (ICDE)*. IEEE, 2023.

[79] Y. Chen, X. Li, G. Cong, Z. Bao, C. Long, Y. Liu, A. K. Chandran, and R. Ellison, "Robust road network representation learning: When traffic patterns meet traveling semantics," in *CIKM*. ACM, 2021, pp. 211–220.

[80] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei, "Deep representation learning for trajectory similarity computation," in *34th IEEE International Conference on Data Engineering, (ICDE), Paris, France, April 16-19, 2018*. IEEE Computer Society, 2018, pp. 617–628.

[81] P. Yang, H. Wang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "T3S: effective representation learning for trajectory similarity computation," in *37th IEEE International Conference on Data Engineering, (ICDE), Chania, Greece, April 19-22, 2021*. IEEE, 2021, pp. 2183–2188.

[82] D. Yao, G. Cong, C. Zhang, and J. Bi, "Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach," in *35th IEEE International Conference on Data Engineering, (ICDE), Macao, China, April 8-11, 2019*. IEEE, 2019, pp. 1358–1369.

[83] Q. Gao, W. Wang, K. Zhang, X. Yang, C. Miao, and T. Li, "Self-supervised representation learning for trip recommendation," *Knowl. Based Syst.*, vol. 247, p. 108791, 2022.

[84] F. Zhou, P. Wang, X. Xu, W. Tai, and G. Trajcevski, "Contrastive trajectory learning for tour recommendation," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 1, pp. 4:1–4:25, 2022.

[85] P. Han, J. Wang, D. Yao, S. Shang, and X. Zhang, "A graph-based approach for trajectory similarity computation in spatial networks," in *KDD*. ACM, 2021, pp. 556–564.

[86] Z. Mao, Z. Li, D. Li, L. Bai, and R. Zhao, "Jointly contrastive representation learning on road network and trajectory," in *CIKM*. ACM, 2022, pp. 1501–1510.

[87] Y. Luo, Q. Liu, and Z. Liu, "STAN: spatio-temporal attention network for next location recommendation," in *WWW*. ACM / IW3C2, 2021, pp. 2177–2185.

[88] Y. Chang, J. Qi, Y. Liang, and E. Tanin, "Contrastive trajectory similarity learning with dual-feature attention," *CoRR*, vol. abs/2210.05155, 2022.

[89] D. Yao, C. Zhang, J. Huang, and J. Bi, "SERM: A recurrent model for next location prediction in semantic trajectories," in *CIKM*. ACM, 2017, pp. 2411–2414.

[90] D. Kong and F. Wu, "HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction," in *IJCAI*. ijcai.org, 2018, pp. 2341–2347.

[91] L. Huang, Y. Ma, S. Wang, and Y. Liu, "An attention-based spatiotemporal LSTM network for next POI recommendation," *IEEE Trans. Serv. Comput.*, vol. 14, no. 6, pp. 1585–1597, 2021.

[92] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, "Trajectory clustering via deep representation learning," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 3880–3887.

[93] X. Liu, X. Tan, Y. Guo, Y. Chen, and Z. Zhang, "CSTRM: contrastive self-supervised trajectory representation model for trajectory similarity computation," *Comput. Commun.*, vol. 185, pp. 159–167, 2022.

[94] D. Lian, Y. Wu, Y. Ge, X. Xie, and E. Chen, "Geography-aware sequential location recommendation," in *KDD*. ACM, 2020, pp. 2009–2019.

[95] J. Manotumruksa, C. Macdonald, and I. Ounis, "A contextual attention recurrent architecture for context-aware venue recommendation," in *SIGIR*. ACM, 2018, pp. 555–564.

[96] K. Sun, T. Qian, T. Chen, Y. Liang, Q. V. H. Nguyen, and H. Yin, "Where to go next: Modeling long- and short-term user preferences for point-of-interest recommendation," in *AAAI*. AAAI Press, 2020, pp. 214–221.

[97] M. Asghari, T. Emrich, U. Demiryurek, and C. Shahabi, "Probabilistic estimation of link travel times in dynamic road networks," in *SIGSPATIAL/GIS*. ACM, 2015, pp. 47:1–47:10.

[98] P. Amirian, A. Basiri, and J. Morley, "Predictive analytics for enhancing travel time estimation in navigation apps of apple, google, and microsoft," in *IWCTS@SIGSPATIAL*. ACM, 2016, pp. 31–36.

[99] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *KDD*. ACM, 2014, pp. 25–34.

[100] H. Zhang, H. Wu, W. Sun, and B. Zheng, "Deeptravel: a neural network based travel time estimation model with auxiliary supervision," in *IJCAI*. ijcai.org, 2018, pp. 3655–3661.

[101] X. Fang, J. Huang, F. Wang, L. Zeng, H. Liang, and H. Wang, "Constgat: Contextual spatial-temporal graph attention network for travel time estimation at baidu maps," in *KDD*. ACM, 2020, pp. 2697–2705.

[102] Y. Shen, C. Jin, J. Hua, and D. Huang, "Ttpnet: A neural network for travel time prediction based on tensor decomposition and graph embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4514–4526, 2022.

[103] S. Xu, R. Zhang, W. Cheng, and J. Xu, "MTLM: a multi-task learning model for travel time estimation," *GeoInformatica*, vol. 26, no. 2, pp. 379–395, 2022.

[104] J. Huang, Z. Huang, X. Fang, S. Feng, X. Chen, J. Liu, H. Yuan, and H. Wang, "Dueta: Traffic congestion propagation pattern modeling via efficient graph learning for ETA prediction at baidu maps," in *CIKM*. ACM, 2022, pp. 3172–3181.

[105] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? estimating travel time based on deep neural networks," in *AAAI*. AAAI Press, 2018, pp. 2500–2507.

[106] H. Yuan, G. Li, Z. Bao, and L. Feng, "Effective travel time estimation: When historical trajectories over road networks matter," in *SIGMOD Conference*. ACM, 2020, pp. 2135–2149.

[107] H. Wang, X. Tang, Y. Kuo, D. Kifer, and Z. Li, "A simple baseline for travel time estimation using large-scale trip data," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 19:1–19:22, 2019.

[108] N. Wu, J. Wang, W. X. Zhao, and Y. Jin, "Learning to effectively estimate the travel time for fastest route recommendation," in *CIKM*. ACM, 2019, pp. 1923–1932.

[109] D. Bernstein, A. Kornhauser *et al.*, "An introduction to map matching for personal navigation assistants," 1996.

[110] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *GIS*. ACM, 2009, pp. 336–343.

[111] T. Osogami and R. Raymond, "Map matching with inverse reinforcement learning," in *IJCAI*. IJCAI/AAAI, 2013, pp. 2547–2553.

[112] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet, "Online map-matching based on hidden markov model for real-time traffic sensing applications," in *ITSC*. IEEE, 2012, pp. 776–781.

[113] T. Hunter, P. Abbeel, and A. Bayen, "The path inference filter: model-based low-latency map matching of probe vehicle data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 507–529, 2013.

[114] G. Hu, J. Shao, F. Liu, Y. Wang, and H. T. Shen, "If-matching: Towards accurate map-matching with information fusion," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 114–127, 2017.

[115] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate GPS trajectories," in *GIS*. ACM, 2009, pp. 352–361.

[116] K. Zhao, J. Feng, Z. Xu, T. Xia, L. Chen, F. Sun, D. Guo, D. Jin, and Y. Li, "Deepmm: Deep learning based map matching with data augmentation," in *SIGSPATIAL/GIS*. ACM, 2019, pp. 452–455.

[117] Z. Shen, W. Du, X. Zhao, and J. Zou, "DMM: fast map matching for cellular data," in *MobiCom*. ACM, 2020, pp. 60:1–60:14.

[118] Z. Jin, J. Kim, H. Yeo, and S. Choi, "Transformer-based map-matching model with limited labeled data using transfer-learning approach," *Transportation Research Part C: Emerging Technologies*, vol. 140, p. 103668, 2022.

[119] L. Jiang, C.-X. Chen, and C. Chen, "L2mm: learning to map matching with deep models for low-quality gps trajectory data," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 3, pp. 1–25, 2023.

[120] N. Wu, W. X. Zhao, J. Wang, and D. Pan, "Learning effective road network representation with hierarchical graph neural networks," in *KDD*. ACM, 2020, pp. 6–14.

[121] M. Wang, W. Lee, T. Fu, and G. Yu, "On representation learning for road networks," *ACM Trans. Intell. Syst. Technol.*, vol. 12, no. 1, pp. 11:1–11:27, 2021.

[122] Y. Chang, E. Tanin, X. Cao, and J. Qi, "Spatial structure-aware road network embedding via graph contrastive learning," in *EDBT*. OpenProceedings.org, 2023, pp. 144–156.

[123] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.

[124] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.

[125] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.

[126] Y. Lin, H. Wan, S. Guo, and Y. Lin, "Contrastive pre-training of spatial-temporal trajectory embeddings," *CoRR*, vol. abs/2207.14539, 2022.

[127] B. Liao, J. Zhang, C. Wu, D. McIlwraith, T. Chen, S. Yang, Y. Guo, and F. Wu, "Deep sequence learning with auxiliary information for traffic prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2018, pp. 537–546.

[128] Z. Cui, R. Ke, and Y. Wang, "Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction," *CoRR*, vol. abs/1801.02143, 2018.

[129] L. Guopeng, V. L. Knoop, and H. van Lint, "Dynamic graph filters networks: A gray-box model for multistep traffic forecasting," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[130] J. Zhang, F. Chen, Z. Cui, Y. Guo, and Y. Zhu, "Deep learning architecture for short-term passenger flow forecasting in urban rail transit," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[131] R. de Medrano and J. L. Aznarte, "A spatio-temporal spot-forecasting framework forurban traffic prediction," *CoRR*, vol. abs/2003.13977, 2020.

[132] L. Liu, J. Chen, H. Wu, J. Zhen, G. Li, and L. Lin, "Physical-virtual collaboration modeling for intra-and inter-station metro ridership prediction," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[133] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 316–324.

[134] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, 2010, pp. 99–108.

[135] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, "Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction," in *AAAI*. AAAI Press, 2019, pp. 5668–5675.

[136] L. Liu, Z. Qiu, G. Li, Q. Wang, W. Ouyang, and L. Lin, "Contextualized spatial–temporal network for taxi origin-destination demand prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3875–3887, 2019.

[137] B. Wang, Y. Lin, S. Guo, and H. Wan, "Gsnet: Learning spatial-temporal correlations from geographical and semantic aspects for traffic accident risk forecasting," in *AAAI*. AAAI Press, 2021, pp. 4402–4409.

[138] M. Kubička, A. Cela, P. Moulin, H. Mounier, and S. Niculescu, "Dataset for testing and training of map-matching algorithms," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1088–1093.

[139] D. Yang, D. Zhang, and B. Qu, "Participatory cultural mapping based on collective behavior data in location-based social networks," *ACM Trans. Intell. Syst. Technol.*, vol. 7, no. 3, pp. 30:1–30:23, 2016.

[140] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *KDD*. ACM, 2011, pp. 1082–1090.

[141] B. Chang, Y. Park, D. Park, S. Kim, and J. Kang, "Content-aware hierarchical point-of-interest embedding model for successive POI recommendation," in *IJCAI*. ijcai.org, 2018, pp. 3301–3307.

[142] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2014.

[143] A. Ziat, E. Delasalles, L. Denoyer, and P. Gallinari, "Spatio-temporal neural networks for space-time series forecasting and relations discovery," in *ICDM*. IEEE Computer Society, 2017, pp. 705–714.

[144] R. de Medrano and J. L. Aznarte, "A spatio-temporal spot-forecasting framework forurban traffic prediction," *CoRR*, vol. abs/2003.13977, 2020.

[145] J. Zhang, F. Chen, Y. Guo, and X. Li, "Multi-graph convolutional network for short-term passenger flow forecasting in urban rail transit," *IET Intelligent Transport Systems*, vol. 14, no. 10, pp. 1210–1217, 2020.

[146] J. Ye, J. Zhao, K. Ye, and C. Xu, "Multi-stgcnet: A graph convolution based spatial-temporal framework for subway passenger flow forecasting," in *2020 International joint conference on neural networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[147] R. de Medrano and J. L. Aznarte, "On the inclusion of spatial information for spatio-temporal neural networks," *Neural Comput. Appl.*, vol. 33, no. 21, pp. 14 723–14 740, 2021.

[148] K. Tian, J. Guo, K. Ye, and C. Xu, "ST-MGAT: spatial-temporal multi-head graph attention networks for traffic forecasting," in *ICTAI*. IEEE, 2020, pp. 714–721.

[149] F. Chen, Z. Chen, S. Biswas, S. Lei, N. Ramakrishnan, and C. Lu, "Graph convolutional networks with kalman filtering for traffic prediction," in *SIGSPATIAL/GIS*. ACM, 2020, pp. 135–138.

[150] C. Shang, J. Chen, and J. Bi, "Discrete graph structure learning for forecasting multiple time series," in *ICLR*. OpenReview.net, 2021.

[151] Y. Wang, H. Yin, H. Chen, T. Wo, J. Xu, and K. Zheng, "Origin-destination matrix prediction via graph convolution: a new perspective of passenger demand modeling," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 1227–1235.

[152] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW*. ACM, 2010, pp. 811–820.

[153] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013.

[154] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G. Sun, "An interactive-voting based map matching algorithm," in *Mobile Data Management*. IEEE Computer Society, 2010, pp. 43–52.

[155] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *CoRR*, vol. abs/1710.10903, 2017.

[156] H. Pei, B. Wei, K. C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," in *ICLR*. OpenReview.net, 2020.

[157] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[158] L. Wu, M. Haynes, A. Smith, T. Chen, and X. Li, "Generating life course trajectory sequences with recurrent neural networks and application to early detection of social disadvantage," in *ADMA*, ser. Lecture Notes in Computer Science, vol. 10604. Springer, 2017, pp. 225–242.

[159] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015.

[160] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2011.

[161] P. J. Huber, "Robust estimation of a location parameter," *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518, 1992.

[162] R. A. Saleh and A. K. M. E. Saleh, "Statistical properties of the log-cosh loss function used in machine learning," *CoRR*, vol. abs/2208.04564, 2022.

[163] R. Koenker and K. F. Hallock, "Quantile regression," *Journal of economic perspectives*, vol. 15, no. 4, pp. 143–156, 2001.

[164] Y. Bengio, P. Y. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[165] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *CoRR*, vol. abs/1807.05118, 2018.

[166] J. Ye, Z. Liu, B. Du, L. Sun, W. Li, Y. Fu, and H. Xiong, "Learning the evolutionary and multi-scale graph structure for multivariate time series forecasting," in *KDD*. ACM, 2022, pp. 2296–2306.

[167] J. Jiang, C. Han, and J. Wang, "Buaa_bigscity: Spatial-temporal graph neural network for wind power forecasting in baidu kdd cup 2022," *arXiv preprint arXiv:2302.11159*, 2023.

[168] R. Jiang, D. Yin, Z. Wang, Y. Wang, J. Deng, H. Liu, Z. Cai, J. Deng, X. Song, and R. Shibasaki, "Dl-traff: Survey and benchmark of deep learning models for urban traffic prediction," in *CIKM*, 2021, pp. 4515–4525.

[169] F. Li, J. Feng, H. Yan, G. Jin, F. Yang, F. Sun, D. Jin, and Y. Li, "Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution," *TKDD*, vol. 17, no. 1, pp. 1–21, 2023.

[170] L. He, X. Liao, W. Liu, X. Liu, P. Cheng, and T. Mei, "Fastreid: A pytorch toolbox for general instance re-identification," *CoRR*, vol. abs/2006.02631, 2020.