LibTraffic: An Open Library for Traffic Prediction (Demo Paper)

Jingyuan Wang[1], Jiawei Jiang[1], Wenjun Jiang[1], Chao Li[1], Wayne Xin Zhao[2]
[1]School of Computer Science and Engineering, Beihang University, Beijing, China
[2]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
{jywang,jwjiang,jwjun,licc}@buaa.edu.cn;batmanfly@gmail.com

ABSTRACT

With the increasing of traffic prediction models, there has become an urgent need to develop a standardized framework to implement and evaluate these methods. This paper presents LibTraffic, a unified, comprehensive, and extensible library for traffic prediction, which provides researchers with a credible experimental tool and a convenient development framework. In this library, we reproduce 42 traffic prediction models and collect 29 spatial-temporal datasets, which allows researchers to conduct comprehensive experiments in a convenient way. To accelerate the development of new models, we design unified model interfaces based on unified data formats, which effectively encapsulate the details of the implementation. To verify the effectiveness of our implementations, we also report the reproducibility comparison results of LibTraffic, and set up a performance leaderboard for the four kinds of traffic prediction tasks. Our library will contribute to the standardization and reproducibility in the field of traffic prediction.

CCS CONCEPTS

• **Information systems → Spatial-temporal systems**.

KEYWORDS

Traffic Prediction, Spatial-temporal System, Reproducibility

ACM Reference Format:
Jingyuan Wang[1], Jiawei Jiang[1], Wenjun Jiang[1], Chao Li[1], Wayne Xin Zhao[2]. 2021. LibTraffic: An Open Library for Traffic Prediction (Demo Paper). In *SIGSPATIAL '21: 29th International Conference on Advances in Geographic Information Systems, November 2-5, 2021, Beijing, China.* ACM, New York, NY, USA, 4 pages.

1 INTRODUCTION

In the area of urban computing, traffic prediction is an important research topic that aims to predict the traffic conditions in the future based on historical traffic data (spatial-temporal data) [4]. Based on specific input and output, traffic prediction can refer to a variety of tasks such as flow prediction, speed prediction, on-demand prediction, and next-location prediction. Traffic prediction plays an important role in many real-world applications such as urban congestion control, route planning, vehicle dispatching and POI recommendation.

In the literature, a number of traffic prediction methods have been proposed, and it has attracted much attention to facilitate the implementation or use of these proposed methods. For example, StreetTraffic is an open-source library that supports the analysis of traffic flow data in real time [2], Flow is a modular reinforcement learning framework for the design and experimentation of neural networks related to autonomous driving control [5], and QarSUMO is a parallel traffic simulator that can provide a large amount of simulated traffic data for road planning and traffic control research [1].

However, based on our literature survey, existing traffic-related libraries are mainly related to traffic data processing and traffic simulation, and there are no open-source libraries for unifying the entire pipeline consisting of data preparation, model design and implementation, and performance evaluation. Besides, there are more and more deep learning models proposed for traffic prediction. We observe that they are usually implemented in very different frameworks and environments, so that it is difficult to reproduce the results of these methods in a unified manner. In particular, deep learning methods are sensitive to the choice of hyperparameters. It has been increasingly difficult to guarantee the effectiveness of new traffic prediction methods and perform the fair evaluation [6]. There is an urgent need to develop a standard framework taking all the details into consideration for traffic prediction.

In this paper, we present a unified, flexible, and comprehensive traffic prediction library named LibTraffic. Our library is implemented based on PyTorch[1], and includes all the necessary steps or components related to traffic prediction into a systematic pipeline. We consider four mainstream tasks, including *traffic speed prediction* (predicting the average speed of vehicles on the road), *traffic flow prediction* (predicting the number of vehicles flowing in or out certain area or road segment), *on-demand service prediction* (predicting the number of requests for a certain region), and *trajectory next-location prediction* (predict where and when the user will go next). We provide various datasets, mechanisms, models, and utilities to support data preprocessing, model instantiation and performance evaluation for the four kinds of tasks.

The main features of LibTraffic can be summarized in three aspects:

• *Unified*: LibTraffic builds a systematic pipeline to implement, use and evaluate traffic prediction models in a unified platform. We design basic spatial-temporal data storage, unified model instantiation interfaces, and standardized evaluation procedure.

• *Comprehensive*: 42 models covering four traffic prediction tasks have been reproduced to form a comprehensive model warehouse. Meanwhile, LibTraffic collects 29 commonly used datasets of different sources and implements a series of commonly used evaluation metrics and strategies for performance evaluation.

[1]https://github.com/pytorch/pytorch

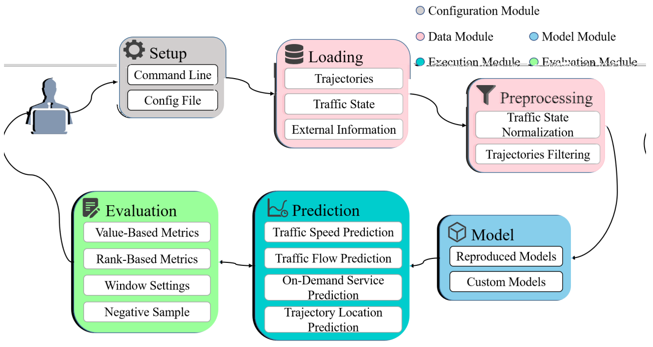

Figure 1: Overview of the LibTraffic library.

• *Extensible*: LibTra□c enables a modular design of di□erent components, allowing users to □exibly insert customized components into the library. Therefore, new researchers can easily develop new models with the support of LibTra□c.

To the best of our knowledge, LibTra□c is the □rst open-source library for tra□c prediction. We believe it provides an important tool to use, explore and develop tra□c prediction models.

2 THE LIBTRAFFIC LIBRARY

The overall framework of LibTra□c is presented in Figure 1, consisting of □ve major modules. As the backbone of LibTra□c, the three core modules are data, model, and evaluation modules, which are respectively responsible for loading and preprocessing datasets, instantiating the prediction models, and evaluating model performance. To run the library, the execution module is responsible for model training and prediction based on the settings of the con□guration module. The aforementioned modules form a systematic pipeline to provide researchers with a credible experimental environment. The following sections will introduce the implementation details of each core module.

2.1 Unified Data Processing Flow

In order to eliminate the unfairness of evaluation caused by di□erent data preparation methods, our data module builds a uni□ed data processing □ow as represented in Figure 2. The entire data □ow involves two special data forms, which are user-oriented and model-oriented respectively. The former form de□nes a uni□ed storage format for spatial-temporal tra□c data (named as *atomic les*) to provide users with a clear data input format, and the latter form de□nes a key-value data structure (named as *batch*) to unify the data interaction between the data module and model module.

Atomic Files. To characterize various kinds of tra□c data, we consider □ve kinds of atomic □les (*i.e.*, the minimum units to format the input). They are listed below: (1) *geographic entity pro le* stores the attributes of geographic entities such as POI, road segment and region; (2) *user entity pro le* stores the attributes of the user entities, who are the participants in tra□c activities; (3) *relation information* stores the relation between entities, such as the road network; (4) *tra c information* stores the tra□c state information or trajectory records; (5) *external information* stores the external information such as weather and events. Based on our literature survey, most of the used tra□c datasets can be formatted with the above □ve kinds of atomic □les. Hence, these atomic □les are useful to unify the input of di□erent tra□c datasets.

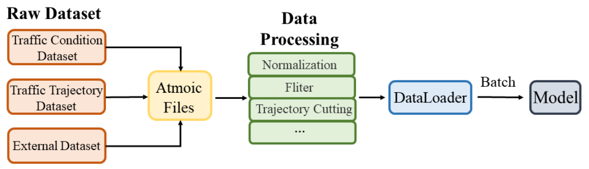

Figure 2: The data processing flow in LibTraffic.

Batch. The *Batch* is a key-value data structure based on the implementation of *python.dict*, where the key is the feature name and value is the corresponding feature tensor in a mini-batch. Such a structure enables the convenient use of feature tensors by referring to corresponding feature names. Although di□erent prediction models use varying features, they all can be stored in the form of *Batch*. Through this data form, LibTra□c can construct uni□ed model interfaces and implement general executor for model training and testing, which is especially useful for developing new models.

Comprehensive Datasets. By surveying the recent literature on tra□c prediction, we select 351 representative or survey papers (more details can be found in Section 2.2). We collected all the open datasets used by these papers, and kept 29 datasets according to the factors of popularity, time span length and data size, which can cover 78.9% papers of our reproduced model list and all the four tasks LibTra□c supports. In order to directly use these datasets in LibTra□c, we have converted all the 29 datasets into the format of atomic □les, and provide the conversion tools for new datasets. Please refer to our GitHub page for dataset statistics, preprocessed copies, and conversion tools at the link https://github.com/LibTra□c/Bigscity-LibTra□c-Datasets.

2.2 Traffic Prediction Model

Once datasets can be prepared in a uni□ed format, we can further implement a variety of tra□c prediction models. Since we have introduced the structure of *Batch* (Section 2.1), we can unify the instantiation interface for tra□c prediction models.

Unified Interfaces. In speci□c, LibTra□c de□nes two standard interfaces: *predict()* and *calculate_loss()*. The interface *predict()* is used in the process of model prediction to return the model prediction results. The interface *calculate_loss()* is used in the process of model training to return the loss value which needs to be optimized. The input of both methods is the internal data representation *Batch*. These interface functions are general to di□erent tra□c prediction models, so that we can implement various models in a highly uni□ed way. When developing a new model, researchers only need to instantiate the above two interfaces to connect with other modules in LibTra□c, while the details of how each of the other parts works can be ignored. This design simpli□es the development process and can accelerate the development of new models.

Comprehensive Models. We conduct a comprehensive review over papers published in the recent □ve years (2016-2020) on 11 top-tier conferences and journals, namely, AAAI, IJCAI, KDD, CIKM, ICDM, WWW, NIPS, ICLR, SIGSPATIAL, IEEE TKDE and IEEE TITS. For AAAI and ICLR, we also collected their papers published in

Table 1: The implemented models in LibTraffic.

Task	Traditional	CNN-based	RNN-based	GCN-based	Attention-based
Traffic flow prediction	AutoEncoder	ST-ResNet, ACFM, STDN	FC-RNN, Seq2Seq	AGCRN, CONVGCN, STSGCN, ToGCN, Multi-STGCnet	ASTGCN, ResLSTM, CRANN, DGCN, DSAN
Traffic speed prediction	AutoEncoder	—	FC-RNN, Seq2Seq	DCRNN, STGCN, GWNET, MTGNN, TGCN, TGCLSTM, ATDM, GTS	GMAN, STAGGCN, HGCN, ST-MGAT
On-Demand service prediction	AutoEncoder	DMVSTNet	FC-RNN, Seq2Seq	CCRNN	STG2Seq
Trajectory next-location prediction	FPMC	—	RNN, ST-RNN, ATST-LSTM, SERM, DeepMove, HST-LSTM, LSTPM, CARA	—	GeoSAN, STAN

2021. In addition, surveys of the past five years and the open-source papers mentioned in these surveys have also been included. Then we only consider the papers with titles or keywords containing "*traffic prediction*", "*traffic forecasting*" and "*next-location prediction*" to construct a paper collection[2] with 351 papers. According to this collection, we further select the models to be included in LibTraffic. We implement 38 methods proposed in these collected papers, from early CNN-based models to recent GCN-based models and hybrid models. Besides, in order to have a good coverage of prediction models, we also implement four shallow baseline models. In total, LibTraffic has reproduced 42 traffic prediction models, categorized Table 1. Due to space limits, we present the details (*e.g.*, references, citation count and venue) about these implemented models in the GitHub link https://github.com/LibTraffic/Bigscity-LibTraffic-Paper/blob/master/reproduced_model.md.

Hyper-parameter Tuning. Considering that hyper-parameter tuning has a great influence on the performance of deep learning models, LibTraffic introduces an automatic hyper-parameter tuning mechanism to reduce the burden of parameter tuning. We implement the automatic hyper-parameter tuning based on the third-party library *Ray Tune* [3], which supports multiple search algorithms such as Grid Search, Random Search and Bayesian Optimization. Users only need to specify the parameters to be adjusted and their search space in a configuration file and select the tuning method. It will sample multiple times from the search space and run the model in a distributed manner, finally automatically saving the best parameter values and corresponding model prediction results.

2.3 Performance Evaluation

With the unified data processing flow and prediction model interfaces, LibTraffic further provides standard evaluation procedures for four traffic prediction tasks, namely traffic speed prediction, traffic flow prediction, on-demand service prediction, and trajectory next-location prediction. For evaluation, the output of the first three tasks is real value, considered as *regression tasks*, while the output of the fourth task is discrete value, considered as *classification tasks*. Hence, we conduct different evaluation settings for the two kinds of tasks. Next, we describe two important aspects related to the evaluation module.

Evaluation Metrics. For regression-based prediction tasks, LibTraffic supports commonly used value-based metrics, which includes MAE, MSE, RMSE, MAPE, Coefficient of Determination (R^2), and Explained variance Score (EVAR). For classification-based tasks, LibTraffic supports commonly used ranking-based metrics, which includes Precision@K, Recall@K, F1-score@K, MRR@K (Mean Reciprocal Rank@K), and NDCG@K (Normalized Discounted Cumulative Gain@K).

Window Setting. For traffic prediction tasks, it is important to set the windows for training and test. For regression-based prediction task, LibTraffic allows users to set varying lengths of input and output time window, and LibTraffic will divide the input data according to the window setting, so that the historical observation data of different time lengths can be used to predict the future traffic states of different time lengths, that is, multi-step prediction. For classification-based task, LibTraffic will split the trajectory according to the window settings. Users can specify the window size and the type of the window (time based or length based) to evaluate the performance of different types of trajectories.

3 EXPERIMENTS WITH LIBTRAFFIC

In this section, we present a series of empirical experiments with the LibTraffic library.

3.1 Reproducibility

In order to verify the correctness of the reproduced models in LibTraffic, we conduct reproducibility experiments by comparing the performance of our implementations and the results reported in the original papers. Note that we only select the papers with open-sourced datasets in order to make the performance comparison.

For regression-based tasks, we conduct experiments with the following models: DCRNN, STGCN, GWNET, ASTGCN, MSTGCN, TGCN, TGC-LSTM, MTGNN, AGCRN, STG2Seq, GMAN, ACFM, STResNet and STSGCN. Note that due to the time steps predicted by different papers are inconsistent, only the results of the three evaluation metrics (MAE, MAPE, RMSE) under the longest time step are shown in the Table 2. For classification-based tasks, we conduct experiments on SERM, DeepMove, LSTPM and STAN, and the experiments' results are shown in Table 3. We will put a blank symbol if there is no result of a certain metric in the original paper. Combining Table 2 and 3, it can be observed that LibTraffic can

[2]https://github.com/LibTraffic/Bigscity-LibTraffic-Paper

achieve similar performance compared to the reported results in the original paper. These results demonstrate the e□ectiveness of the reproduced models in LibTra□c.

Table 2: Reproducibility comparison for regression-based prediction task.

Model	Dataset	MAE		RMSE		MAPE	
		original	ours	original	ours	original	ours
DCRNN	METR_LA	3.599	3.6	7.620	7.59	10.51%	10.50%
STGCN	PeMSD7(M)	3.608	3.57	6.763	6.77	8.87%	8.69%
GWNET	METR_LA	3.587	3.53	7.421	7.37	10.76%	10.01%
ASTGCN	PEMSD4	21.108	21.8	33.802	32.82	—	—
MSTGCN	PEMSD4	19.334	22.73	31.150	35.64	—	—
TGCN	SZ-taxi	2.769	2.7889	4.100	4.0141	—	—
TGC-LSTM	Loop-Seattle	2.770	2.57	4.122	4.63	6.90%	6.01%
MTGNN	METR_LA	3.467	3.49	7.217	7.23	9.90%	9.87%
AGCRN	PEMSD4	19.509	19.83	32.296	32.26	13.23%	12.97%
STG2Seq	TAXIBJ	9.692	10.219	16.900	17.241	15.21%	13.80%
GMAN	PEMS_BAY	2.115	1.86	4.321	4.32	4.80%	4.31%
ACFM	TAXIBJ	—	—	15.286	15.4	—	—
STResNet	TAXIBJ	—	—	16.937	16.69	—	—
STSGCN	PEMSD4	22.460	21.19	34.903	33.65	15.95%	13.90%

Table 3: Reproducibility comparison for classification-based prediction task.

Model	Dataset	Recall@1		Recall@5		Recall@10	
		original	ours	original	ours	original	ours
SERM	NY	0.254	0.084	0.451	0.172	0.543	0.202
DeepMove	Foursquare	0.145	0.138	0.284	0.276	—	—
LSTPM	Foursquare	0.156	0.135	0.337	0.279	0.409	0.330
STAN	NYC	—	—	0.467	0.470	0.596	0.581

3.2 Performance Leaderboard

Typically, a research paper does not need to consider all the related methods as baselines. Instead, it should be compared with the most relevant or competitive methods. Hence, we further conduct experiments to derive the performance leaderboard for each task.

Experimental Setting. For tra□c speed and □ow prediction tasks, we choose two sensor-based datasets respectively, where METR-LA and PEMS_BAY datasets are used for speed prediction, and PEMSD4 and PEMSD8 are used for □ow prediction. For the above datasets, we adopt a 5-minute window for aggregating tra□c data. For multi-step tra□c forecasting, we use one-hour historical data to predict the next-hour condition, adopt MAE, RMSE and MAPE as evaluation metrics. For tra□c demand prediction task, we choose two grid-based datasets (NYCTaxi20150103 and NYCBike20160708), which are aggregated into 30-minutes and 1-hour windows respectively. For multi-step tra□c forecasting, we use 12-hour historical data to predict the condition in the next three hours, and adopt MAE, RMSE and R^2 as evaluation metrics. Due the essence of regression-based prediction, the models aimed for tra□c □ow prediction can be also applied to demand prediction tasks. Therefore, we conduct the experiment with the reproduced □ow prediction models. For the three tasks above, the datasets are split in a chronological order with 70% for training, 10% for validation and 20% for testing. For tra□c trajectory next-location prediction task, we choose two datasets: Foursquare-TKY and Instagram, and we remove unpopular POIs with fewer than 3 check-ins. We use 24-hour time window to split the trajectory and remove trajectories with fewer than 5 check-ins. Inactive users with fewer than 3 trajectories are also □ltered. For each user, we divide her/his trajectories into three parts with a ratio of 6:2:2, namely training set, validation set and test set. We train the model with the training set, optimize the model with validation set and adopt Recall@5, MRR@5, and NDCG@5 to evaluate.

Results and Analysis. For each task, we obtain the performance of the comparison methods at each predicted time step for a dataset, and average the results for performance ranking. Table 4 presents the top □ve positions for each task in the performance leaderboard. In general, we can see that the top □rst models of the four tasks are MTGNN, AGCRN, ACFM and LSTPM, respectively. Specially, for regression-based tasks, graph neural network-based models outperform others, and MTGNN and AGCRN both introduce an adaptive graph adjacency matrix generation method to replace the pre-de□ned graph structure. While, CNN-based methods ACFM and STResNet performed well for grid-based datasets. Besides, for classi□cation-based tasks, LSTPM outperforms the others and attention-based or semantic-enhanced methods also achieved good performance. We will continue to update this performance leaderboard.

Table 4: Top five positions in the performance leaderboard.

Task	1	2	3	4	5
Tra□c Speed	MTGNN	GWNET	DCRNN	STGCN	GMAN
Tra□c Flow	AGCRN	STSGCN	CONVGCN	DGCN	ASTGCN
Tra□c Demand	ACFM	STResNet	ASTGCN	CONVGCN	AGCRN
Trajectory Next-Location	LSTPM	SERM	DeepMove	STAN	ST-RNN

4 CONCLUSION

This paper presented a uni□ed, comprehensive, and extensible library for tra□c prediction, called LibTra□c. It collected 29 spatial-temporal datasets and reproduced 42 tra□c prediction models, covering four mainstream tasks of tra□c prediction. We conducted the experiments to verify the implementation e□ectiveness andenend2Tra:tiorasli