BasicTS: An Open Source Fair Multivariate Time Series Prediction Benchmark

Yubo Liang[1,2], Zezhi Shao[1,2], Fei Wang[1(✉)], Zhao Zhang[1], Tao Sun[1], and Yongjun Xu[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
{liangyubo20g,shaozezhi19b,wangfei,zhangzhao2021,suntao,xyj}@ict.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. Multivariate Time Series (MTS) is ubiquitous in the real world, and its prediction plays a vital role in a wide range of applications. Recently, many researchers have made persistent efforts to design powerful models. For example, Spatial-Temporal Graph Neural Networks (STGNNs) have become increasingly popular MTS prediction methods due to their state-of-the-art performance. However, we found there exists much unfairness in the comparison of the performance of existing models, which may prevent researchers from making correct judgments. Meanwhile, researchers usually have to build training pipelines that are complex and error-prone when designing new models, which further obstacles the quick and deep innovation in the MTS prediction field. In this paper, we first analyze the sources of unfairness and then propose a fair and easy-to-use benchmark, BasicTS, to address the above two issues. On the one hand, for a given MTS prediction model, BasicTS evaluates its ability based on rich datasets and standard pipelines. On the other hand, BasicTS provides users with flexible and extensible interfaces to facilitate convenient designing and exhaustive evaluation of new models. In addition, based on BasicTS, we provide performance revisits of several popular MTS prediction models. The proposed benchmark is publicly available at https://github.com/zezhishao/BasicTS.

Keywords: Multivariate time series prediction · Unfairness · Benchmark

1 Introduction

Multivariate Time Series (MTS) contains time series from multiple correlated variables and exists in many real-world systems. Accurate MTS prediction fuels a wide range of services related to intelligent transportation, financial investment, and environmental protection. It helps people to make better decisions. Thus, MTS prediction has remained an enduring topic in both academia and industry.

Z. Shao—Project leader.

A. Gainaru et al. (Eds.): Bench 2022, LNCS 13852, pp. 87–102, 2023.
https://doi.org/10.1007/978-3-031-31180-2_6

Despite the significant progress, we find that the evaluation and comparison of existing models are not fair enough, which may lead researchers to make wrong judgments and thus hind innovation in the field of MTS prediction. Specifically, after an exhaustive technical review of existing works, we summarize the sources of unfairness into three levels: data level, model level, and evaluation level.

Data Level:
Unfairness caused by the lack of richness of the datasets. Different datasets are often heterogeneous, i.e., datasets have different physical characteristics, dynamics, and so on. Therefore, the same model may have different performances on different types of datasets. Thus, using only a specific type of dataset for comparison may lead to unfair results.
Unfairness caused by data pre-processing. Different normalization methods (e.g., max-min normalization, z-score normalization) may affect the model's performance. Therefore, if different models adopt different pre-processing approaches, the comparison of their performance results is unfair.

Model Level:
Unfairness caused by different pipelines. Pipeline controls many details of the training process. Since each researcher tends to construct their own model pipeline, it may bring an unfair comparison of results.
Unfairness caused by hyper-parameters settings. In deep learning-related prediction models, the hyper-parameters have a significant impact on the final performance, e.g., learning rate, weight decay, random seeds, etc. For example, we find that in some works [1,2], the performance of important baselines, such as DCRNN [3] and Graph WaveNet [4], is surprisingly poor, this may be caused by unreasonable hyper-parameter settings. Thus, different settings of hyper-parameters may lead to an unfair comparison of results.

Evaluation Level:
Unfairness caused by different ways of calculating the evaluation metrics. Common evaluation metrics for MTS prediction problems include MAE, MAPE, RMSE. Although they have strict mathematical definitions, the implementation details may vary, such as the way of handling outliers, and mini-batch computations [5]. These differences can cause significant deviations from test results and actual performance, thus leading to an unfair comparison of results.
Unfairness caused by different ways of evaluation. For example, in the field of MTS prediction, the metrics of **horizon x** denotes the error metrics **at** the x-th prediction time step, while many researchers make mistakes and calculate the average of the error metrics over 0-x prediction time step, which results in a significant reduction in error and thus significant unfairness.

In order to solve the above unfairness problems and fairly evaluate the performance of a given model, we propose a fair and easy-to-use open-source benchmark for MTS prediction, named BasicTS. Specifically, BasicTS provides an exhaustive and fair evaluation of a given model based on a unified pipeline and rich datasets. In addition, to make it easier for researchers to use, BasicTS provides a set of rich and extensible interfaces that allows users to focus on model

design and ignore the building of the training and evaluating pipeline, enabling rapid development and comprehensive evaluation. Finally, based on BasicTS, we present a performance review of popular deep learning-based MTS predicting methods, to provide researchers with a solid reference.

Our contributions are summarized as follows:

- We designed a benchmark named BasicTS for solving the unfair comparison problem of MTS prediction models. For a given MTS prediction model, BasicTS utilizes a unified pipeline to perform an exhaustive evaluation of its capabilities based on rich datasets.
- We designed a set of rich and extensible interfaces in BasicTS, which can help researchers quickly design and evaluate their own models and be free from the hassle of building complex pipelines.
- Based on BasicTS, we provide a fair performance comparison of existing popular MTS prediction models to provide researchers with a solid reference and thus inspire innovations.

2 Related Works

In this section, we list the existing benchmarks related to time series prediction.

GluonTS [6] is an open-source benchmark designed by Amazon that focuses on time series prediction. However, it cannot handle datasets with pre-defined graphs, which limits its usability for STGNN-related models. FOST is a spatio-temporal prediction framework designed by MSRA. Compared to GluonTS, it adds a GNN model to ensure its ability to handle data with pre-defined graphs. However, FOST lacks interfaces for hyper-parameter settings, and it can only make predictions but cannot evaluate the prediction results, which makes it difficult to guarantee the fairness of this benchmark. In addition, FOST contains only three models (RNN, CNN, GNN) and has not designed interfaces to add new models. Also, its form of input data is fixed, which significantly limits its extensibility. LibCity [7] is a library specifically focused on traffic-related problems, which aims to provide experimental tools for researchers. However, it is not designed for benchmarking and only focuses on traffic-related data, ignoring many other real-world MTS prediction problems.

Compared with existing works, BasicTS is the first work that provides unified pipelines and rich datasets for benchmarking given MTS prediction models, and provides users with extensible and easy-to-use interfaces for quickly designing and evaluating new models.

3 Benchmark Building

In the Introduction, we analyze the factors that may lead to unfairness. In this section, we will explore our ideas to solve the above unfairness problems and propose the specific implementation of BasicTS.

3.1 Design Thoughts

In this part, we demonstrate the design thoughts of BasicTS, which aims to address the critical unfairness issues discussed in the Introduction and provide extensibility for users to enable users adding their models and datasets.

Unfairness. We propose the following solutions to the factors that lead to unfairness in the field of MTS prediction:

Data Level. For unfairness caused by the lack of richness of the datasets, we used rich and heterogeneous datasets. Specifically, BasicTS currently includes ten datasets, including traffic speed datasets (METR-LA, PEMS-BAY), traffic flow datasets (PEMS03, PEMS04, PEMS07, PEMS08), electricity, solar-energy, exchange-rate, and Beijing air quality. In particular, in addition to datasets that include a pre-defined graph indicating spatial dependency, the latter four datasets, which do not contain a pre-defined graph, can help to evaluate the model's capability more comprehensively. For unfairness caused by data pre-processing, we adopted a uniform data pre-processing process, which takes Z-Score normalization as default.

Model Level. For unfairness caused by different pipelines, we use an identical, standard, extensible pipeline to avoid the unfairness problem caused by different training pipelines. For unfairness caused by hyper-parameter settings, we have provided interfaces that allow flexible parameter settings and carefully tuned the parameters of all existing models in BasicTS.

Evaluation Level. We use unified evaluation metrics and pipelines to ensure the fairness of the evaluation. In addition, to measure the model's performance at different prediction lengths, users can evaluate the performance of the model at any time step less than the length of the prediction.

Extensibility. BasicTS provides researchers with rich, easy-to-use, and extensible interfaces to configure the standard Pipeline and functions built in BasicTS. Specifically, for the convenience of researchers, a unified configuration file is designed to allow users to configure all parameters, such as dataloader, environment, and parameters to be optimized. Users can configure it by simply editing them at the string level as if they were filling out a form. In addition, the unified configuration file imports the model to be evaluated and its runner (optional), which can be designed at will by simply following the standard input and output interfaces designed by BasicTS.

The unified configuration profile and extensible interface design allow users to ignore the construction of the training process and focus on the design of the model, enabling rapid iteration and effective innovation.

3.2 Implementation of BasicTS

The specific implementation of BasicTS is shown in Fig. 1. Among them, users communicate with BasicTS through a unified configuration file. In this part, we will describe the implementation of each module in detail.

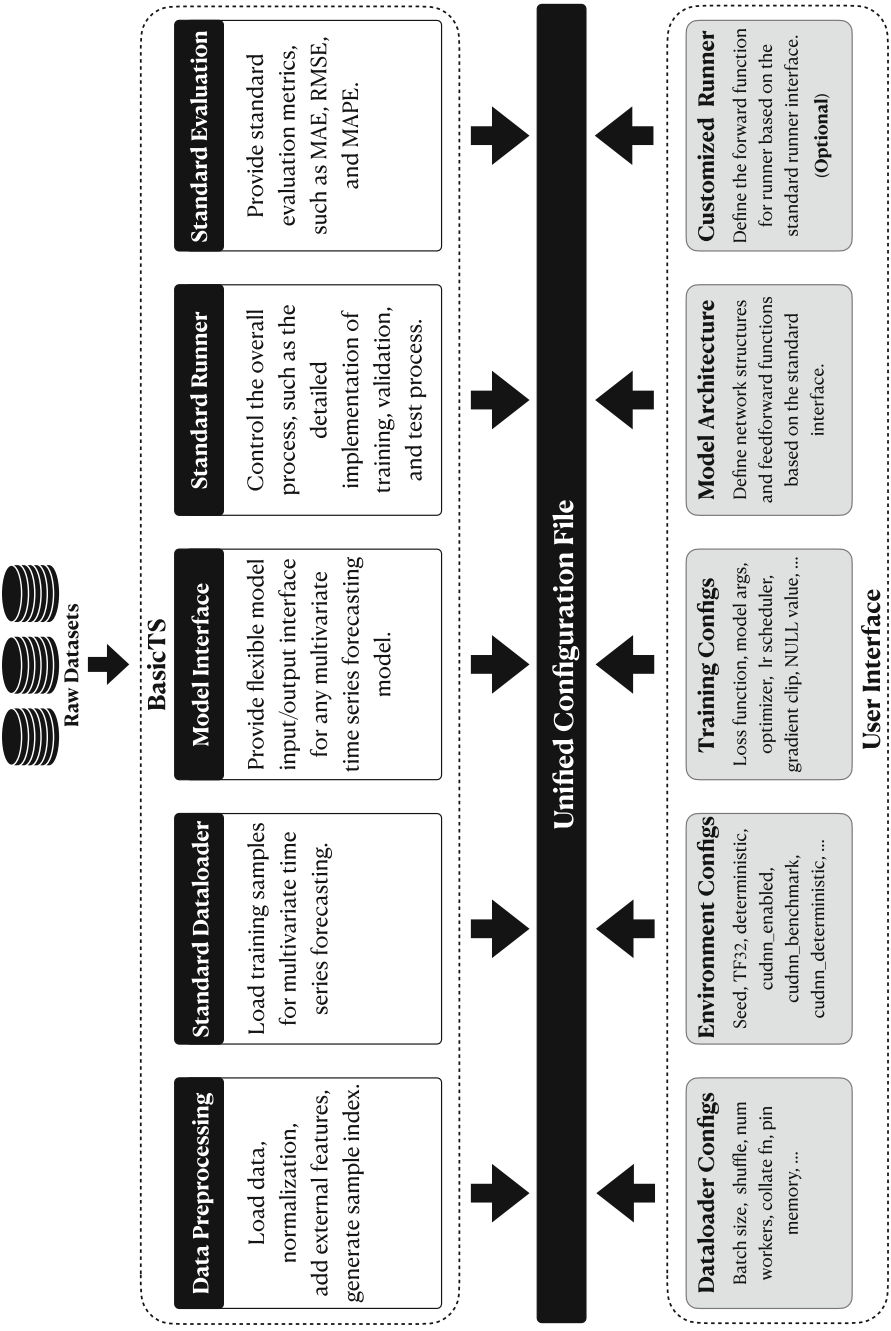

Fig. 1. BasicTS components.

Data Pre-processing: The data pre-processing module aims to generate a unified data file for the model. Specifically, the data pre-processing module first loads the original data for pre-processing (e.g., normalization) and adds additional features (e.g., time features *Time of Day*, *Day of Week*). Then, training samples are obtained by sliding windows of length $P + F$ over the time series, where the first P time slices are historical data and the subsequent F time slices are future data. In particular, to improve flexibility and efficiency, BasicTS stores the index of the sample instead of the sample itself.

Standard Dataloader: Benefiting from the uniform data storage format generated by data pre-processing module, this module can read any dataset in a standard mini-batch way.

Model: In this module, BasicTS specifies the standard input and output interfaces. The input interface contains common parameters such as historical data, epochs number, iteration number, and so on. The model aims to return prediction values. By following the model interfaces specified by BasicTS, users are free to design arbitrary models, which fully guarantees the extensibility of BasicTS.

Runner: Runner controls the entire training, validation, and testing process, such as data loader construction, model optimization, evaluation methods, model saving and loading, log saving, and other details. BasicTS includes a built-in standard runner to ensure fairness. Users can adjust the standard runner by modifying the parameters in the configuration file, such as learning rate, weight decay coefficients, and so on. In addition, we also allow users to customize the runner in an inherited way.

Evaluation: This module is designed to evaluate the results produced by the runner. BasicTS currently provides implementations of three widely-used evaluation metrics, MAE, MAPE, and RMSE. BasicTS's standard runner evaluates the results on **Horizon@3,6,12, overall** using the incoming metrics, thus avoiding the unfairness caused by algorithmic evaluation.

User Interface: Users communicate with BasicTS by configuring the unified configuration file, which is a python file that maintains an EasyDict object that allows the user to edit at the string level, like filling out a form. For example, users can set BATCH_SIZE to set the batch size of the Dataloader and CFG.TRAIN.OPTIM.TYPE to "Adam" to use Adam as the model's optimizer. the unified configuration file allows users to configure almost all parameters related to Dataloader, such as environment variables, training parameters, and so on. In addition, users can import their own designed model structure and custom runner (optional) into the unified configuration file.

4 Evaluation

In this section, we first introduce the setup of our experiments, and then we illustrate the unfair phenomenon mentioned in the Introduction through experiments. Finally, we provide a fair performance revisit of existing popular MTS prediction models.

4.1 Experimental Setup

Datasets: We conducted experiments on ten commonly used MTS prediction datasets: PEMS03, PEMS04, PEMS07, PEMS-08, PEMS-BAY, METR-LA, Electricity, solar-energy, exchange-rate, and Beijing air quality.

The information of these datasets is shown in Table 1. Traffic-related predictions, such as traffic flow prediction and traffic speed prediction, are the most common issues of MTS prediction. Among them, PEMS03, PEMS04, PEMS07, and PEMS08 are traffic flow datasets, while PEMS-BAY and METR-LA are traffic speed datasets. These datasets contain a pre-defined graph indicating the spatial dependency between traffic sensors.

However, the MTS problem has a wide range of applications in many fields. Therefore, we also include four datasets from different areas. They are electricity and solar-energy for energy, exchange-rate for economics, and Beijing air quality dataset for environmental protection. Since there are no spatial dependencies among multiple time series in the applications of these domains, none of these four datasets contain pre-defined graphs.

Table 1. Information of datasets used in BasicTS.

Dataset	Length	Variants	Sample Rate	Time Span	Application
PEMS03	26208	358	5 min	3 months	traffic flow
PEMS04	16992	307	5 min	2 months	traffic flow
PEMS07	28224	883	5 min	3 months	traffic flow
PEMS08	17856	170	5 min	6 months	traffic flow
PEMS-BAY	52116	325	5 min	6 months	traffic speed
METR-LA	6850	207	5 min	4 months	traffic speed
Electricity	2208	336	60 min	3 months	electricity
solar-energy	52560	137	10 min	1 year	energy
exchange-rate	7588	8	1 day	20 years	economics
Beijing air quality	6000	7	6 h	1500 days	environment

Models: In this part, we briefly introduce the MTS prediction baselines included in BasicTS. Particularly, we choose MTS prediction models that contain official public code, which helps researchers to make a quick and accurate comparison and reproduction.

- HI [8]: Historical Inertia (HI) model adopts the most recent historical data points in input time series as the prediction results.
- LSTM [9]: Long Short-Term Memory (LSTM) network with fully connected hidden units is a well-known network architecture that is powerful in capturing sequential dependency.
- DCRNN [3]: Diffusion Convolutional Recurrent Neural Network (DCRNN) models the traffic flow as a diffusion process. It replaces the fully connected layer in GRU with a diffusion convolutional layer to form a new Diffusion Convolutional Gated Recurrent Unit (DCGRU).

- Graph WaveNet [4]: Graph WaveNet stacks Gated TCN and GCN layer by layer to jointly capture the spatial and temporal dependencies.
- STGCN [10]: Spatial-Temporal Graph Convolutional Network (STGCN) integrate graph convolution (spatial dimension) and 2D gated temporal convolution (temporal dimension) to model the correlations in MTS data.
- StemGNN [1]: Spectral Temporal Graph Neural Network (StemGNN) takes the advantage of both inter-series correlations and temporal dependencies by modeling them jointly in the spectral domain.
- MTGNN [11]: MTGNN extends Graph WaveNet through the mix-hop propagation layer in the spatial module, the dilated inception layer in the temporal module, and a more delicate graph learning layer.
- DGCRN [12]: DGCRN models the dynamic graph and designs a novel Dynamic Graph Convolutional Recurrent Module (DGCRM) to capture the spatial-temporal pattern in a seq2seq architecture.
- GTS [5]: GTS learns a graph structure among multiple time series and forecasts them simultaneously with DCRNN.
- AGCRN [13]: Adaptive Graph Convolutional Recurrent Network (AGCRN) captures node-specific spatial and temporal correlations in MTS based on two modules, i.e., node adaptive parameter learning and data-adaptive graph generation modules.
- STNorm [14]: STNorm refines the high-frequency component and the local component from the MTS data based on the proposed temporal normalization and spatial normalization, respectively.
- D^2STGNN [15]: D^2STGNN decouples the diffusion and inherent signals built in MTS data to achieve more precise modeling, and features a dynamic graph learning module for the dynamic characteristics of traffic networks.

Metrics: We evaluated all models by three most widely used metrics in MTS prediction, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Squared Error (MSE). In addition, we compared the performance of these methods on the error metrics at the 3, 6, 12, and overall prediction time steps, which is shown in the Horizon@3, @6, @12, and overall, respectively.

Experimental Environment: All models are trained on Intel(R) Xeon(R) Gold 5217 CPU @ 3.00 GHz, 128G RAM computing server, equipped with NVIDIA RTX 3090 graphics cards.

4.2 Experimental Results

In this section, we will experimentally demonstrate the unfairnesses mentioned in the Introduction.

Unfairness Caused by Lack of Richness of the Datasets. Different datasets often have different properties, e.g., different distributions, different dynamics, etc. Therefore, even the same model tends to show different performances on different datasets. Here, we select two typical models, GTS and

Table 2. Comparison of DCRNN and GTS performance on different datasets.

Datasets	Models	@Horizon 3			@Horizon 6			@Horizon 12			Overall (12 Horizon)		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
METR-LA	DCRNN	2.67	5.16	6.86%	3.07	6.29	8.42%	3.57	7.56	10.37%	3.04	6.26	8.33%
	GTS	2.75	5.28	7.13%	3.14	6.33	8.70%	3.60	7.46	10.42%	3.10	6.29	8.54%
PEMS-BAY	DCRNN	1.31	2.80	2.73%	1.66	3.81	3.75%	1.98	4.64	4.73%	1.60	3.74	3.61%
	GTS	1.36	2.91	2.85%	1.72	3.86	3.88%	2.05	4.62	4.87%	1.65	3.77	3.73%
PEMS03	DCRNN	14.16	24.61	14.21%	15.41	27.01	15.07%	17.31	30.05	16.71%	15.37	26.92	15.10%
	GTS	13.93	23.96	14.02%	15.27	26.12	15.35%	17.35	29.11	17.23%	15.24	26.08	15.24%
PEMS04	DCRNN	18.53	29.61	12.71%	19.65	31.37	13.45%	21.67	34.19	15.03%	19.71	31.43	13.54%
	GTS	19.27	30.46	13.33%	20.86	32.78	14.68%	23.52	36.31	17.03%	20.91	32.86	14.77%
PEMS07	DCRNN	19.45	31.39	8.29%	21.18	34.42	9.01%	24.14	38.84	10.42%	21.20	34.43	9.06%
	GTS	20.00	31.87	8.45%	22.11	35.02	9.39%	25.49	39.77	10.96%	22.08	35.07	9.40%
PEMS08	DCRNN	14.16	22.20	9.31%	15.24	24.26	9.90%	17.70	27.14	11.13%	15.26	24.28	9.96%
	GTS	14.50	22.97	9.23%	15.77	25.08	10.09%	18.02	28.25	11.74%	15.82	25.13	10.18%

DCRNN, for performance comparison on six different traffic-related datasets we mentioned above.

As shown in Table 2, The two models have different performances on different datasets. DCRNN performs better on METR-LA, PEMS-BAY, PEMS04, PEMS07, and PEMS08; however, GTS performs better on PEMS03. Therefore, we introduce 10 datasets in multiple domains to comprehensively measure the performance of a model on each dataset. In particular, six traffic datasets contain a predefined graph to describe spatial associations; the other four datasets do not contain predefined graphs. This helps to comprehensively measure the ability of the model to handle different datasets.

Unfairness Caused by Data Pre-processing. For most machine-learning-related models, it is essential to perform data pre-processing on the raw data. Among the pre-processing methods, normalization is the most common means, which helps to improve the efficiency of gradient descent and enables models to obtain better results. Common normalization methods include Z-score normalization and max-min normalization. Here, we choose three models, Graph WaveNet, STGCN, AGCRN to compare the effects of different data pre-processing methods on the results. The experiments were conducted on the PEMS-BAY dataset.

Table 3. Effect of different data pre-processing methods on MTS prediction.

Methods	Min-max Normailization			Z-score Normalization		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
GraphWaveNet	1.56	3.57	3.49%	1.59	3.69	3.52%
STGCN	1.66	3.72	3.70%	1.63	3.73	3.69%
AGCRN	1.69	3.91	3.81%	1.63	3.78	3.73%

As shown in Table 3, The same model may show very different results when using different data pre-processing methods. Therefore, when comparing results,

it is crucial to ensure that all models use the same data pre-processing methods. We provide a convenient normalization processing interface in the data preparation stage mentioned above, which can fully guarantee fairness in this aspect.

Unfairness Caused by Different Pipelines. Different model pipelines may likewise lead to an unfair comparison of results. For example, Whether or not to add gradient clipping to the training pipeline will have a great impact on the result. Here, we tested the effect of adding gradient clipping to the MTGNN and STNorm's training pipeline on the exchange-rate dataset, shown in Table 4.

Table 4. Effect of different pipeline on MTS prediction.

Methods	Add Gradient clipping			Not Add Gradient clipping		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
MTGNN	0.0133	0.0227	7.05%	0.0130	0.0205	5.45%
STNorm	0.0068	0.0116	1.81%	0.0070	0.0118	2.63%

As shown in Table 4, adding gradient clipping to the training pipeline has a huge impact on the results. Furthermore, there are also many other details about pipeline construction, which can also affect the results. Therefore, different pipelines often bring significant unfairnesses. As described in the Benchmark Building, we use the same pipeline for all models to circumvent the possible unfair comparison of results.

Unfairness Caused by Hyper-Parameters Setting. Hyper-parameters setting is an integral part of determining a model's effect, and there are considerable works on this.

There is a wide variety of hyper-parameters, including optimizer [16,17], weight-decay [18], batch_size [19,20], and so on. Many of them can significantly impact the algorithm's performance. Here, we set up two experiments to show the effect of the hyper-parameter settings on the model.

Optimizer. Optimizer is one of the most important hyper-parameters in deep learning models. It refers to the method of finding the optimal deep neural network parameters through gradient descent, which determines the efficiency and stability of gradient learning optimization methods. Here, we test the effects of Adam [16], Adagrad [17], and SGD [21] optimizers on the performance of model STGCN on PEMS-BAY. The result is shown in Table 5.

Weight Decay. The strong fitting ability of neural networks may lead to overfitting. Therefore, it is often necessary to take measures to improve the generalization ability of neural networks. Weight decay is one of the most common regularization methods, which improves the generalization ability of neural networks by introducing a discount factor when the parameters are updated. Therefore, the coefficient of weight decay is one of the most important hyper-parameters of the deep learning model. Here, we set the weight decay coefficients

Table 5. Effect of different optimizers on the performance of STGCN on PEMS-BAY.

Optimizer	@Horizon 3			@Horizon 6			@Horizon 12			Overall (12 Horizon)		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Adam	1.35	2.86	2.86%	1.69	3.83	3.85%	2.00	4.56	4.74%	1.63	3.73	3.69%
Adagrad	1.46	3.09	3.07%	1.88	1.98	4.78%	2.41	5.51	5.93%	1.86	4.26	4.26%
SGD	1.67	3.35	3.56%	2.10	4.63	4.57%	2.79	6.33	6.45%	2.12	4.78	4.70%

to 0.00001,0.0001,0.001 to test the effect of the weight decay coefficient on the performance of model STGCN on PEMS-BAY. The result is shown in Table 7.

Table 6. Effect of different weight-decay coefficients on the performance of STGCN on PEMS-BAY.

Coefficient	@Horizon 3			@Horizon 6			@Horizon 12			Overall (12 Horizon)		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
0.00001	1.37	2.70	2.88%	1.71	3.92	3.83%	2.03	4.70	4.66%	1.65	3.83	3.67%
0.0001	1.35	2.86	2.86%	1.69	3.83	3.85%	2.00	4.56	4.74%	1.63	3.73	3.69%
0.001	1.59	3.35	3.63%	1.92	4.24	4.54%	2.29	5.17	5.47%	1.88	4.21	4.44%

As shown above, hyper-parameters show a considerable impact on the performance of the model. Therefore, reasonable parameter adjustment is an important part of ensuring the fairness of the comparison of the performance of models. However, the optimal hyper-parameters of different models are often different, and the adjustment of hyper-parameters still depends largely on artificial experience. To this end, we provided interfaces that allows flexible parameter settings. Also, we have carefully tuned the parameters of all existing models in BasicTS to make them optimal (Table 6).

4.3 Review

In this subsection, we review 11 models on METR-LA, PEMS-BAY, PEMS03, PEMS04, PEMS07, PEMS08, and Electricity, solar-energy, exchange-rate, and Beijing air quality. In particular, some models require a pre-defined adjacency matrix as input, thus these models will not work on the latter four datasets. We divide reviews into three categories: traffic speed datasets, traffic flow datasets, and datasets that does not contain pre-defined graph.

For a fair comparison, we follow the dataset division in previous works. The ratio of training, validation, and test sets for the PEMS-BAY dataset is 7 : 1 : 2, while the ratio for other datasets is 6 : 2 : 2. We aim to predict the future time series with a length of 12, i.e., $F = 12$, on all datasets. We compared the performance of these methods on the 3rd, 6th, and 12th time slots and the average 12 time slots, which are shown in the **@3**, **@6**, **@12**, and **@overall** columns, respectively. The results of the review are shown in Table 7, Table 8, Table 9.

Table 7. Review of MTS prediction methods on dataset which doesn't contain predefined graph.

Datasets	Methods	@Horizon 3			@Horizon 6			@Horizon 12			Overall (12 Horizon)		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Electricity	HI	92.42	167.00	70.16%	92.58	167.05	70.46%	92.79	167.21	70.91%	92.58	167.07	70.43%
	AGCRN	22.88	50.02	41.30%	24.49	54.16	48.90%	27.25	59.80	52.57%	23.87	53.00	10.16%
	StemGNN	21.43	46.80	35.08%	22.02	49.87	40.00%	26.06	56.99	47.59%	22.75	49.80	39.52%
	MTGNN	16.78	36.91	48.17%	18.43	42.61	51.32%	20.51	48.34	56.27%	18.19	42.04	50.77%
	STNorm	18.94	40.77	39.10%	21.73	47.70	51.66%	24.62	55.04	66.98%	21.32	47.46	49.49%
Solar Energy	HI	7.20	9.65	376.10%	7.20	9.65	376.10%	7.20	9.65	376.10%	7.20	9.65	376.10%
	AGCRN	1.48	2.61	101.08%	2.02	3.39	136.36%	2.76	4.50	158.56%	1.98	3.45	125.68%
	StemGNN	1.74	2.83	128.85%	2.26	3.62	161.17%	2.88	4.58	183.25%	2.21	3.63	151.21%
	MTGNN	1.35	2.41	70.70%	1.81	3.06	107.07%	2.56	4.09	178.68%	1.80	3.13	109.25%
	STNorm	0.56	1.58	59.70%	0.77	2.10	101.69%	1.13	2.84	169.64%	0.77	2.14	96.57%
Exchange Rate	HI	0.0092	0.0151	1.18%	0.0092	0.0151	1.18%	0.0092	0.0151	1.18%	0.0092	0.0151	1.18%
	AGCRN	0.0060	0.0088	4.83%	0.0082	0.0127	2.29%	0.0106	0.0168	2.05%	0.0082	0.0130	3.33%
	StemGNN	0.1521	0.1991	179.07%	0.1511	0.1974	199.18%	0.1534	0.1998	192.41 %	0.1549	0.2022	169.47%
	MTGNN	0.0133	0.0227	7.05%	0.0167	0.0273	8.30%	0.0184	0.0300	7.34%	0.0164	0.0273	7.70%
	STNorm	0.0048	0.0081	0.69%	0.0068	0.0112	1.01%	0.0098	0.0156	2.77%	0.0068	0.0116	1.81%
Beijing Air Quality	HI	30.20	57.99	99.54%	30.27	58.03	99.60%	30.24	58.02	99.43%	30.23	58.01	99.47%
	AGCRN	30.16	53.60	119.24%	31.41	55.53	130.92%	32.63	58.77	139.12%	30.20	54.14	126.12%
	StemGNN	27.02	48.07	143.93%	27.09	48.55	211.92%	26.64	48.55	129.88%	26.65	48.48	151.80%
	MTGNN	21.68	42.02	78.52%	25.66	46.39	129.62%	26.24	47.83	120.64%	23.66	44.39	100.39%
	STNorm	20.69	39.07	92.66%	23.64	42.63	102.89%	24.26	44.65	99.14%	21.99	41.05	100.28%

Table 8. Review of MTS prediction methods on traffic speed datasets.

Datasets	Methods	@Horizon 3			@Horizon 6			@Horizon 12			Overall (12 Horizon)		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
METR-LA	HI	6.80	14.21	16.72%	6.80	14.21	16.72%	6.80	14.20	10.15%	6.80	14.21	16.72%
	Graph WaveNet	2.69	5.15	6.96%	3.08	6.21	8.47%	3.53	7.30	10.15%	3.04	6.15	8.31%
	DCRNN	2.67	5.16	6.86%	3.07	6.29	8.42%	3.57	7.56	10.37%	3.04	6.26	8.33%
	AGCRN	2.88	5.57	7.72%	3.26	6.61	9.17%	3.67	7.60	10.74%	3.20	6.50	9.00%
	STGCN	2.76	5.31	7.20%	3.16	6.36	8.72%	3.62	7.45	10.43%	3.12	6.30	8.58%
	StemGNN	2.96	5.77	7.90%	3.46	6.96	9.79%	4.11	8.32	12.25%	3.43	6.93	9.70%
	GTS	2.75	5.28	7.13%	3.14	6.33	8.70%	3.60	7.46	10.42%	3.10	6.29	8.54%
	MTGNN	2.71	5.22	6.89%	3.07	6.23	8.27%	3.51	7.28	9.90%	3.04	6.17	8.15%
	STNorm	2.82	5.55	7.48%	3.19	6.59	9.00%	3.56	7.47	10.51%	3.12	6.45	8.77%
	STID	2.79	5.53	7.64%	3.16	6.57	9.30%	3.53	7.51	10.78 %	3.10	6.45	9.01%
	DGCRN	2.61	5.02	6.57%	2.99	6.07	7.90%	3.45	7.27	9.49%	2.96	6.05	7.79%
	D^2STGNN	2.56	4.90	6.52%	2.90	5.90	7.88%	3.34	7.02	9.63%	2.87	5.88	7.79%
PEMS-BAY	HI	3.06	7.05	6.85%	3.06	7.04	6.84%	3.05	7.03	6.83%	3.05	7.05	6.84%
	Graph WaveNet	1.30	2.80	2.69%	1.65	3.75	3.65%	1.97	4.58	4.63%	1.59	3.69	3.52%
	DCRNN	1.31	2.80	2.73%	1.66	3.81	3.75%	1.98	4.64	4.73%	1.60	3.74	3.61%
	AGCRN	1.37	2.93	2.95%	1.70	3.89	3.88%	1.99	4.64	4.72%	1.63	3.78	3.73%
	STGCN	1.35	2.86	2.86%	1.69	3.83	3.85%	2.00	4.56	4.74%	1.63	3.73	3.69%
	StemGNN	1.44	3.12	3.08%	1.93	4.38	4.54%	2.57	5.88	6.55%	1.91	4.46	4.54%
	GTS	1.36	2.91	2.85%	1.72	3.86	3.88%	2.05	4.62	4.87%	1.65	3.77	3.73%
	MTGNN	1.34	2.84	2.80%	1.67	3.79	3.74%	1.97	4.55	4.57%	1.60	3.70	3.57%
	STNorm	1.34	2.88	2.82%	1.67	3.83	3.75%	1.96	4.52	4.62%	1.60	3.71	3.60%
	STID	1.30	2.81	2.73%	1.62	3.72	3.68%	1.89	4.40	4.47%	1.55	3.62	3.51%
	DGCRN	1.29	2.80	2.74%	1.63	3.80	3.75%	1.95	4.58	4.64%	1.58	3.71	3.61%
	D^2STGNN	1.25	2.65	2.62%	1.58	3.63	3.57%	1.86	4.37	4.44%	1.52	3.55	3.50%

Table 9. Review of MTS prediction methods on traffic flow datasets.

Datasets	Methods	@Horizon 3			@Horizon 6			@Horizon 12			Overall (12 Horizon)		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
PEMS03	HI	32.46	49.78	30.58%	32.45	49.76	30.59%	32.44	49.75	30.63%	32.45	49.76	30.60%
	Graph WaveNet	13.37	23.04	13.90%	14.51	25.29	14.85%	16.16	27.91	16.12%	14.48	25.19	14.67%
	DCRNN	14.16	24.61	14.21%	15.41	27.01	15.07%	17.31	30.05	16.71%	15.37	26.92	15.10%
	AGCRN	14.22	25.02	13.40%	15.47	27.28	14.43%	17.09	28.78	16.43%	15.41	27.15	14.76%
	STGCN	14.71	25.19	14.41%	15.66	26.99	15.38%	17.47	29.80	17.55%	15.73	27.03	15.44%
	StemGNN	14.16	24.33	14.40%	15.76	26.98	15.32%	18.50	30.94	18.10%	15.87	27.10	15.60%
	GTS	13.93	23.96	14.02%	15.27	26.12	15.35%	17.35	29.11	17.23%	15.24	26.08	15.24%
	MTGNN	13.71	23.04	14.84%	14.87	25.94	15.12%	16.50	28.76	16.88%	14.80	25.65	15.04%
	STNorm	14.23	24.05	13.98%	15.45	26.54	14.49%	17.08	29.42	15.73%	15.34	26.33	14.56%
	STID	17.51	28.48	12.00%	18.29	29.86	12.46%	19.58	31.79	13.38%	18.29	29.82	12.49%
	DGCRN	13.46	23.92	14.23%	14.67	26.36	15.13%	16.41	29.02	16.71%	14.61	26.15	15.10%
	D^2STGNN	13.42	23.11	13.71%	14.71	25.61	14.73%	16.62	28.69	16.64%	14.72	25.61	14.70%
PEMS04	HI	42.33	61.64	29.90%	42.35	61.66	29.92%	42.38	61.67	29.96%	42.35	61.66	29.92%
	Graph WaveNet	18.00	28.83	13.64%	18.96	30.33	14.23%	20.53	32.54	15.41%	18.97	30.32	14.26%
	DCRNN	18.53	29.61	12.71%	19.65	31.37	13.45%	21.67	34.19	15.03%	19.71	31.43	13.54%
	AGCRN	18.52	29.79	12.31%	19.45	31.45	12.82%	20.64	33.31	13.74%	19.36	31.28	12.81%
	STGCN	18.74	29.84	12.93%	19.64	31.34	13.27%	21.12	33.53	14.22%	19.63	31.32	13.32%
	StemGNN	19.48	30.74	13.84%	21.40	33.46	15.85%	24.90	38.29	19.50%	21.61	33.80	16.10%
	GTS	19.27	30.46	13.33%	20.86	32.78	14.68%	23.52	36.31	17.03%	20.91	32.86	14.77%
	MTGNN	18.65	30.13	13.32%	19.48	32.02	14.08%	20.96	34.66	14.96%	19.50	32.00	14.04%
	STNorm	18.28	29.70	12.28%	18.92	31.12	12.71%	20.20	32.91	13.43%	18.96	30.98	12.69%
	STID	17.51	28.48	12.00%	18.29	29.86	12.46%	19.58	31.79	13.38%	18.29	29.82	12.49%
	DGCRN	17.88	29.12	12.25%	18.86	30.92	12.85%	20.20	33.20	13.80%	18.81	30.82	12.80%
	D^2STGNN	17.44	28.48	11.91%	18.20	29.91	12.29%	19.31	31.68	12.99%	18.15	29.80	12.25%
PEMS07	HI	49.02	71.15	22.73%	49.04	71.18	22.75%	49.06	71.21	22.79%	49.03	71.18	22.75%
	Graph WaveNet	18.69	30.69	8.02%	20.26	33.37	8.56%	22.79	37.11	9.73%	20.25	33.32	8.63%
	DCRNN	19.45	31.39	8.29%	21.18	34.42	9.01%	24.14	38.84	10.42%	21.20	34.43	9.06%
	AGCRN	19.31	31.68	8.18%	20.70	34.52	8.66%	22.74	37.94	9.71%	20.64	34.39	8.74%
	STGCN	20.33	32.73	8.68%	21.66	35.35	9.16%	24.16	39.48	10.26%	21.71	35.41	9.25%
	StemGNN	19.74	32.32	8.27%	22.07	36.16	9.20%	26.20	42.32	11.00%	22.23	36.46	9.20%
	GTS	20.00	31.87	8.45%	22.11	35.02	9.39%	25.49	39.77	10.96%	22.08	35.07	9.40%
	MTGNN	19.23	31.15	8.55%	20.83	33.93	9.30%	23.60	38.10	10.10%	20.94	34.03	9.10%
	STNorm	19.15	31.70	8.26%	20.63	35.10	8.84%	22.60	38.65	9.60%	20.52	34.85	8.77%
	STID	18.31	30.39	7.72%	19.59	32.90	8.30%	21.52	36.29	9.15%	19.54	32.85	8.25%
	DGCRN	18.57	30.49	7.82%	20.12	33.43	8.45%	22.31	37.04	9.44%	20.05	33.32	8.45%
	D^2STGNN	18.56	30.52	7.79%	20.10	33.15	8.41%	22.30	36.73	9.40%	20.05	33.08	8.42%
PEMS08	HI	36.65	50.44	21.60%	36.66	50.45	21.63%	36.68	50.46	21.68%	36.66	50.45	21.63%
	Graph WaveNet	13.72	21.71	8.80%	14.67	23.50	9.49%	16.15	25.85	10.74%	14.67	23.47	9.52%
	DCRNN	14.16	22.20	9.31%	15.24	24.26	9.90%	17.70	27.14	11.13%	15.26	24.28	9.96%
	AGCRN	14.51	22.87	9.34%	15.66	25.00	10.34%	17.49	27.93	11.72%	15.65	24.99	10.17%
	STGCN	14.95	23.48	9.87%	15.92	25.36	10.42%	17.65	28.03	11.34%	15.98	25.37	10.43%
	StemGNN	14.49	23.02	9.73%	15.84	25.38	10.78%	18.10	28.77	12.50%	15.91	25.44	10.90%
	GTS	14.50	22.97	9.23%	15.77	25.08	10.09%	18.02	28.25	11.74%	15.82	25.13	10.18%
	MTGNN	14.30	22.55	10.56%	15.25	24.41	10.54%	16.80	26.96	10.90%	15.31	24.42	10.70%
	STNorm	14.44	22.68	9.22%	15.53	25.07	9.94%	17.20	27.86	11.30%	15.54	25.01	10.00%
	STID	13.28	21.66	8.62%	14.21	23.57	9.24%	15.58	25.89	10.33%	14.20	23.49	9.28%
	DGCRN	13.47	21.87	8.85%	14.44	23.77	9.44%	15.90	26.35	10.50%	14.43	23.75	9.40%
	D^2STGNN	13.24	21.83	8.47%	14.19	23.98	9.09%	15.50	26.43	9.90%	14.20	23.95	9.10%

5 Conclusion

In this paper, we propose a fair, standard, and open-source benchmark for multivariate time series prediction, named BasicTS, to address the unfairnesses in the comparison of MTS prediction models. Given a model, BasicTS evaluates it based on rich datasets, standard training pipeline, and standard evaluation, to give a fair performance validation. Furthermore, BasicTS provides users with flexible and extensible interfaces to facilitate quick designing and fair evaluation of new MTS prediction models. Last but not least, we also provide a fair performance review of several popular MTS prediction models based on BasicTS.

6 Future Works

This paper explores and evaluates the unfairness of the MTS prediction and proposes a framework dedicated to the MTS prediction problem. In the future, we will continue this research in three aspects:

1. MTS prediction problems contain a wide variety of methods and data forms. We plan to add more datasets and models into BasicTS. We will also conduct more experiments on these datasets.
2. The MTS prediction models also include some long-time prediction models. We plan to add more long-time prediction models into BasicTS.
3. With the development of machine learning, auto hyperparameter optimization techniques are beginning to be used more and more abundantly. We plan to add auto hyperparameter optimization technology into our benchmark, which can help researchers to find optimal parameters for deep learning models conveniently.

References

1. Cao, D., et al.: Spectral temporal graph neural network for multivariate time-series forecasting. In: Advances in Neural Information Processing Systems, vol. 33, pp. 17766–17778 (2020)
2. Li, M., Zhu, Z.: Spatial-temporal fusion graph neural networks for traffic flow forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4189–4196 (2021)
3. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: data-driven traffic forecasting. arXiv preprint arXiv:1707.01926 (2017)
4. Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C.: Graph wavenet for deep spatial-temporal graph modeling. arXiv preprint arXiv:1906.00121 (2019)
5. Shang, C., Chen, J., Bi, J.: Discrete graph structure learning for forecasting multiple time series. arXiv preprint arXiv:2101.06861 (2021)
6. Alexandrov, A., et al.: GluonTS: probabilistic and neural time series modeling in Python. J. Mach. Learn. Res. **21**(116), 1–6 (2020)
7. Wang, J., Jiang, J., Jiang, W., Li, C., Zhao, W.X.: Libcity: an open library for traffic prediction. In: Proceedings of the 29th International Conference on Advances in Geographic Information Systems, pp. 145–148 (2021)

8. Cui, Y., Xie, J., Zheng, K.: Historical inertia: a neglected but powerful baseline for long sequence time-series forecasting. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 2965–2969 (2021)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875 (2017)
11. Gao, J., et al.: MTGNN: multi-task graph neural network based few-shot learning for disease similarity measurement. Methods **198**, 88–95 (2022)
12. Li, F., et al.: Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. ACM Trans. Knowl. Discov. Data (TKDD) **17**, 1–21 (2021)
13. Bai, L., Yao, L., Li, C., Wang, X., Wang, C.: Adaptive graph convolutional recurrent network for traffic forecasting. In: Advances in Neural Information Processing Systems, vol. 33, pp. 17804–17815 (2020)
14. Deng, J., Chen, X., Jiang, R., Song, X., Tsang, I.W.: ST-Norm: spatial and temporal normalization for multi-variate time series forecasting. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 269–278 (2021)
15. Shao, Z., et al.: Decoupled dynamic spatial-temporal graph neural network for traffic forecasting. arXiv e-prints, pp. arXiv-2206 (2022)
16. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
17. Lydia, A., Francis, S.: Adagrad-an optimizer for stochastic gradient descent. Int. J. Inf. Comput. Sci. **6**(5), 566–568 (2019)
18. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
19. Smith, S.L., Kindermans, P.-J., Ying, C., Le, Q.V.: Don't decay the learning rate, increase the batch size. arXiv preprint arXiv:1711.00489 (2017)
20. Radiuk, P.M.: Impact of training set batch size on the performance of convolutional neural networks for diverse datasets (2017)
21. Amari, S.-I.: Backpropagation and stochastic gradient descent method. Neurocomputing **5**(4–5), 185–196 (1993)