Scalable Transformer for High Dimensional Multivariate Time Series Forecasting

Xin Zhou
Monash University
Melbourne, Victoria, Australia
xin.zhou@monash.edu

Weiqing Wang*
Monash University
Melbourne, Victoria, Australia
teresa.wang@monash.edu

Wray Buntine
VinUniversity
Hanoi, Vietnam
wray.b@vinuni.edu.vn

Shilin Qu
Monash University
Melbourne, Victoria, Australia
qshilin@foxmail.com

Abishek Sriramulu
Tymestack
Melbourne, Victoria, Australia
abishek.sriramulu1@monash.edu

Weicong Tan
Monash University
Melbourne, Victoria, Australia
weicong.tan@monash.edu

Christoph Bergmeir†
University of Granada
Granada, Spain
christoph.bergmeir@monash.edu

Abstract

Deep models for Multivariate Time Series (MTS) forecasting have recently demonstrated significant success. Channel-dependent models capture complex dependencies that channel-independent models cannot capture. However, the number of channels in real-world applications outpaces the capabilities of existing channel-dependent models, and contrary to common expectations, some models underperform the channel-independent models in handling high-dimensional data, which raises questions about the performance of channel-dependent models. To address this, our study first investigates the reasons behind the suboptimal performance of these channel-dependent models on high-dimensional MTS data. Our analysis reveals that two primary issues lie in the introduced noise from unrelated series that increases the difficulty of capturing the crucial inter-channel dependencies, and challenges in training strategies due to high-dimensional data. To address these issues, we propose STHD, the **S**calable **T**ransformer for **H**igh-**D**imensional Multivariate Time Series Forecasting. STHD has three components: a) Relation Matrix Sparsity that limits the noise introduced and alleviates the memory issue; b) ReIndex applied as a training strategy to enable a more flexible batch size setting and increase the diversity of training data; and c) Transformer that handles 2-D inputs and captures channel dependencies. These components jointly enable STHD to manage the high-dimensional MTS while maintaining computational feasibility. Furthermore, experimental results show STHD's considerable improvement on three high-dimensional datasets: Crime-Chicago, Wiki-People, and Traffic. The source code and dataset are publicly available [1].

CCS Concepts

• **Information systems** → **Data mining**; **Spatial-temporal systems**.

Keywords

Multivariate Time Series Forecasting, High-dimensional Time Series, Forecasting Accuracy

ACM Reference Format:
Xin Zhou, Weiqing Wang, Wray Buntine, Shilin Qu, Abishek Sriramulu, Weicong Tan, and Christoph Bergmeir. 2024. Scalable Transformer for High Dimensional Multivariate Time Series Forecasting. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3627673.3679757

1 Introduction

Multivariate Time Series (MTS) forecasting involves predicting future values for multiple variables simultaneously based on historical time series data. This technique shows promise and applicability across various domains and is gaining attention due to its wide applications in forecasting natural disasters [2, 7, 16, 19, 21, 27, 29, 41, 62], government management [4, 36, 37, 63, 78, 82], monitoring human health [1, 28, 50, 59, 67, 75, 76], and decision-making [13, 20, 26, 30, 32, 51, 64, 69, 74, 77, 81], etc. MTS differs from univariate time series by consisting of multiple time-dependent variates, each contributing to the dynamics of the dataset. Recently, the increasing amount of large-scale MTS data has led to two significant developments: the extension of time dimensions (long-time dimension) and a substantial increase in the number of channels (high-dimensional) [2], each bringing new challenges to statistical forecasting methods. Statistical forecasting methods [3, 22, 38], are often designed according

*Corresponding author.
†Also serve for Monash University, Melbourne, Australia.

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679757

[1] https://github.com/xinzzzhou/ScalableTransformer4HighDimensionMTSF.git
[2] **'channel', 'dimension', and 'variate' are interchangeable in this work.**

to the characteristics of the data, which usually will become very complex for large-scale MTS data.

Deep Neural Network (DNN) models have shown great success in many areas, because of their ability to model non-linear transformations, versatility, and scalability. Given the critical assumption for time series data that future values depend on historical values, mainstream research mainly focuses on the long-time data [42, 66, 72] and most existing works simply model high-dimensional data with channel (or dimension)-independent approaches [40]. This focus has led to significant advances in understanding and modeling temporal dynamics over long-time periods. Transformer-based methods emerge as a promising technique for enhancing temporal analysis, offering significant performance in data mining and predictive It is a mature DNN model that is now popular for time series [11, 42, 83, 84]. PatchTST [42], iTransformer [37], and Crossformer [78]) are leading transformer-based methods that show great performance in MTS forecasting. They represent two categories of MTS forecasting technologies: channel-independent and channel-dependent models. These terms refer to how various dimensions in MTS data are treated relative to each other during the training process. However, same with the other mainstream DNN models, they focus more on long-time forecasting instead of high-dimensional MTS forecasting. In this work, we focus on using modern transformer architectures to address the gap for high-dimensional MTS forecasting. Our motivation is underscored by existing works [54, 78, 80], which emphasize the importance of learning relations among variables for prediction (i.e., channel-dependent modeling).

Applying channel-dependent transformers to high-dimensional MTS forecasting presents specific challenges, including learning complex segment relationships from across all channels, and coping with increased memory and computational complexity. One of the primary challenges we observed and reported in this paper is that some existing channel-dependent models underperform channel-independent models on high-dimensional data, contrary to common expectations that accounting for dimension relationships enhances forecasting accuracy. Specifically, on three datasets: Crime-Chicago, Wiki-People, and Traffic, one state-of-the-art channel-dependent transformer, Crossformer, is on average 19% less accurate than the state-of-the-art channel-independent transformer, PatchTST. Another challenge we observe is the computational overhead in modeling correlations among all the segments in a multitude of interacting channels. Existing work can only model the time relations within individual series [42], or for channel relations within individual segment [78]. A naïve solution is to learn the attention map in the shape of $(M \times S) \times (M \times S)$, where S is the number of subseries split from each series, and M is the total number of series. However, learning a huge attention map directly imposes a significant computational burden. Furthermore, the scale of high dimensions strains memory and processing power. It limits existing channel-dependent models from being designed with large values for parameters such as batch size and embedding size. Take the Crossformer as an example. The batch sampling size is $b \times M$, where b is the batch size. For the high-dimensional data, M is large, increasing the memory and processing power usage, making it M times larger than in usual cases. To ensure the algorithm runs successfully, a compromise solution is to limit the tuning range of parameters (e.g., setting smaller batch sizes), but this leads to the sacrifice in model performance.

Our research aims to bridge the above gaps, and at the same time, to retain the performance as a channel-dependent model. Therefore, we propose a novel approach to scalably manage the channels, their comprehensive dependencies, and parameter settings along with the extended time dimensions inherent in high-dimensional MTS. To the best of our knowledge, we are the first to use the modern Transformer architecture for high-dimensional MTS forecasting, with the ability to explore information from dependencies between channels.

Our key contributions are listed as follows: First, we conduct a detailed analysis to understand the reasons behind the suboptimal performance of channel-dependent models on high-dimensional MTS data, identifying the main issue is the introduced noise by possibly many unrelated channels. Second, we propose STHD, the **S**calable **T**ransformer for **H**igh-**D**imensional Multivariate Time Series Forecasting. To ensure efficiency, STHD designs a 2-D transformer with a sparsed attention map to be learned enabled by DeepGraph [53] which sparsifies the relations between a large number of channels [49]. Apart from the improved learning efficiency, enforcing sparsity in the 2-D transformer also reduces introduced noise, and reduces memory and processing power consumption. Third, channel-dependent MTS models need large numbers of parameters, causing memory issues. Our proposed "ReIndex" strategy optimizes computational resources and increases the diversity of shuffled training samples, ultimately aiding STHD in better generalization. Finally, STHD presents a significant improvement in high-dimensional MTS forecasting on three real-world high-dimensional datasets: Crime-Chicago, Wiki-People, and Traffic, which have higher dimensions compared with the datasets used in the existing works. In our experimental evaluations, STHD achieved an average 8.91% performance improvement on the three datasets.

2 Related Work

2.1 Channel-independent Model

A channel-independent model treats each channel as an individual series, to forecast with just the given series. For the data whose channels may not have significant dependencies, channel-independent modeling simplifies the process. Statistical methods in this field are mostly global models or local univariate models, adapting to the features of data. One of the most prevalent approaches is Auto-Regressive Integrated Moving Average(ARIMA) [3], which incorporates autoregression, differencing, and moving averages, providing a detailed understanding of time series dynamics. Another fundamental statistical method is Exponential Smoothing (ETS) [22] applies weighting factors that decrease exponentially over time. ETS is widely used for smoothing time series data to identify trends and patterns. However, the computation speed and memory of the above methods limit their scalability for long, and high-dimensional data analysis.

DNNs make a significant advance in handling sequential time series data. Recurrent Neural Networks (RNNs), particularly Long

Short-Term Memory (LSTM), represent channel-independent paradigms [43, 66]. LSTM learns long-term dependencies, making them suitable for modeling complex temporal dynamics in MTS forecasting. Their ability to remember information over extended periods and to handle large-scale data address the limitation of statistical methods. Recent advances in deep learning for forecasting, focus on Convolutional Neural Networks (CNNs) [34, 52, 58, 79] and attention-based models [5, 8, 17, 24, 31, 48, 56, 57, 60, 65, 71]. DeepGLO [44], designed for univariate time series forecasting, can learn both global and local views of all series through CNNs. Time Series with Transformer (TST) [73] is the first work that applies transformer to MTS forecasting. Informer [83] focuses on efficient transformer forecasting. FEDformer focuses on the time-frequency analysis of individual series. PatchTST [42] divides series into different patches (segments) and through the Transformer framework learns dynamic time relationships. TSMixer [11] replaces the attention mechanism with linear layers to improve computation efficiency. TimesNet [61] splits time series into different frequency domains to capture comprehensive temporal dynamics. LLM-based models [23, 25, 85] first tokenize series, then extract information in pretrained language models, which is time-consuming for fine-tuning than transformer's training. The above methods, especially transformers, can handle long sequences more effectively and provide more flexibility in capturing complex temporal relations, so we intend to design a transformer-based model to solve high-dimensional MTS forecasting.

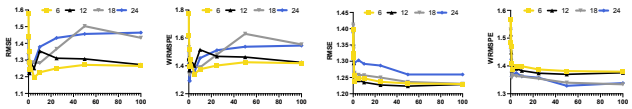
2.2 Channel-dependent Model

The channel-dependent model recognizes and leverages the dependencies among channels, that is to forecast with both target series and auxiliary series. Recognizing these dependencies can lead to more accurate and explainable forecasts. The fundamental statistical method is Vector Auto-Regression (VAR) [47], which captures linear dependencies among multiple time series. Despite a simple and effective framework for MTS analysis, VAR might not perform well with non-linear data. Besides, some statistical methods are used for channel dependency analysis in different aspects [49]: Correlation Analysis, Granger Causality, Mutual Information, Maximum Likelihood Estimation, Naive Transfer Entropy, Optimal Causation Entropy, and Thouless-Anderson-Palmer. The above methods are hard to scale for high-dimensional data due to slow computation.

DNNs can be channel-dependent in linear and non-linear ways. Representative linear methods are Linear, Normalization Linear (NLinear), and Decomposed Linear (DLinear) [72]. They are linear model implemented in the Deep-learning framework. DLinear applies a linear model on the decomposed elements in different frequency domains of time series. With this, DLinear can precisely capture the seasonality, trend, and residual components. DeepVAR [43] is an RNN-based time series model with a Gaussian copula process output model. TST [73] is a pre-trained model for time series representation learning. STEP [46] learns both the spatial and temporal patterns based on TST. Crossformer [78] develops a transformer-based model to learn both relations cross time and cross dimension, separately. Graph Neural Networks (GNNs) are wildly used in forecasting [6, 9, 10, 12, 14, 15, 33, 35, 39, 45, 55, 63]. Ye et al. [68] model time series with a hierarchical graph structure to capture the scale-specific correlations among series. FourierGNN [70] define time series data as a hypervariable graph structure and a Fourier learning method. The evolving nature of time series necessitates that the above models be adapted to dynamic patterns and incorporate diverse data characteristics. There is a growing trend towards hybrid models that blend statistical and machine learning techniques. ADLNNs [49] first apply statistical methods to the sparse correlation matrix, then apply GNNs to assist forecasting. They can capture the dynamic co-evolutions and use the information from other channels to improve the forecast of the target one. The main challenge lies in effectively modeling the complex and possible non-linear dependencies. This often requires complex model design, thus leading to high computational memory requirements.

3 Initial Analysis

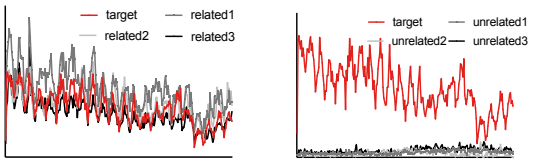
In this section, we conduct exploratory experiments and data analysis on a high-dimensional dataset: Crime-Chicago, a precursor to the model development phase. We employ Crossformer [78] and iTransformer [37], state-of-the-art channel-dependent transformer models, as our pre-test models. We start with a randomly selected series from the dataset as the target series. Then, we use the last 20% as the test sub-series. Subsequently, we sample different proportions of top K correlated series with target series as training data, where $K \in \{0, 1, 5, 10, 15, 57, 114, 288, 576, 1154\}$, representing the proportions in {0%, 0.1%, 0.5%, 0.9%, 1.3%, 5%, 10%, 25%, 50%, 100%} respectively. Figure 1 shows the performance of training with the above data for Crossformer and iTransformer (iTransfor for short in Figure 1).



(a) Crossformer (b) Crossformer (c) iTransfor (d) iTransfor

Figure 1: Result of using different proportions of training data on Crossformer and iTransformer. The x-axis describes proportions from 0% to 100%. The y-axis in (a) and (c) represents RMSE on Crossformer and iTransformer (iTransfor for short), and the y-axis in (b) and (d) represents WRMSPE on Crossformer and iTransformer. Lines in yellow, black, grey, and blue represent the horizon of {6,12,18,24} months.

Please note that, both metrics in Figure 1 are metrics of errors which means that they both are the less the better. From Figure 1a-1b, we find a sharp metric downtrend at the very beginning and then was followed by an obvious rise. From Figure 1c-1d, we find when the proportion equals 50%, the performance is the best. This shows that for both channel-dependent models, for the target series, utilizing some series achieves the best performance compared with utilizing all series or none at all. We guess the reason is that utilizing 100% of the series as training data across channel-dependent models, which have full connections to all series, can potentially introduce noise stemming from unrelated connections. That means, a proper number of related series, may introduce useful patterns to forecasting. To confirm this, we perform the data analysis in Figure 2.



(a) target series with top 3 related series

(b) target series with top 3 unrelated series

Figure 2: Visualisation of the target series in red, and its related series/non-related series in gray and black.

Figure 2a and Figure 2b compare the visualization of the same target series (in red) with its related series Figure 2a and unrelated series Figure 2b. We can observe that related series show similar trends with target series, while unrelated series show different and irregular trends. Overall, in high-dimensional data, introducing unrelated patterns, and unrelated series can lead to suboptimal performance.

4 Scalable Transformer STHD

Problem Formulation. We are solving Multivariate Time Series Forecasting with a channel-dependent model for high-dimensional data. Following the common practice of existing works [42, 78], we use the sliding window to split each series into S subseries, with each containing an input and a horizon, when training, the loss computes a little overlap of series, which is similar to the common practice in NLP. Given the historical observation set of time series $\mathcal{X} = \{X_1, X_2, ..., X_M\} \in \mathbb{N}^{M\times T\times(1+K)}$, which consists of M series with 1 target series, and K related series for each, all of them are in T time points. For example, in the Wiki-People prediction dataset, the series $\mathcal{X}$ is the traffic of each article, and K articles that show a similar temporal pattern. The output is the future τ time points' observations (Horizon) $\mathcal{Y} = \{Y_1, Y_2, ..., Y_M\} \in \mathbb{N}^{M\times\tau}$ for M series. After the sliding window process, the number of windows $S = \lfloor \frac{T-1}{L} \rfloor + 2$, where L is the window length, the stride is set to 1. Finally, the input $\mathcal{X}' \in \mathbb{N}^{M\times S\times L\times(1+K)}$, the output $\mathcal{Y}' \in \mathbb{N}^{M\times S\times\tau}$.

Overview of STHD. The supervised learning pipeline is shown in Figure 3. First, STHD starts with the Relation Matrix Sparsity module, which refines the input data by focusing on the most relevant inter-channel relationships. According to relation matrix $\Gamma \in \mathbb{R}^{M\times M}$, STHD gets K related series with 1 target series as input. This not only enhances the model's ability to capture channel dependencies but also alleviates the memory issue commonly encountered in handling high-dimensional data. This content is introduced in Section 4.1. Then, the refined data passes through the ReIndex module, which ensures efficient data processing and enhances training diversity, ultimately aiding in better model generalization. We introduce this content in Section 4.2. Finally, the 2-D Transformer, introduced in Section 4.3, with its capability to handle 2-D inputs, utilizes the processed and optimized data to capture the complex dependencies between channels over time. STHD predicts future $\mathcal{Y}'$. Collectively, these modules are optimized for computational efficiency, making STHD viable for high-dimensional MTS forecasting.

4.1 Relation Matrix Sparsity

Based on the assumption in Section 3 that not all inter-series relationships contribute equally to forecasting accuracy, STHD uses Pearson Correlation, an efficient method, to capture the extent of series co-variates. The impact of this component is twofold: it improves forecasting accuracy by reducing the influence of noise and irrelevant information and improves computational efficiency with DeepGraph as it allows the model to focus on the related series instead of all the series.

Referring to ADLNN [49], which demonstrates the effectiveness of correlation matrix sparsity, we use correlation matrix sparsity to get K related series. However, the computation is very slow, especially for high-dimensional MTS data. To facilitate efficient relation computation and manage the relationships within the dataset, we use DeepGraph [53], a structure used for computing relations, to model high-dimensional MTS data by formulating individual series as a node and relation with another series as an edge. It enables the dynamic construction of a sparse correlation matrix by identifying and mapping significant inter-series relationships. The fast computation of pairwise correlation matrices in the DeepGraph framework is attributed to a combination of advanced data structures, robust scalability, and parallelization. Central to its efficiency are optimized data structures, tailored to handle complex and large datasets efficiently with full control over RAM usage. This is complemented by the parallelization and integration of Cython [3], which significantly boosts performance. To get the sparse correlation matrix $\mathbf{\Gamma}$, for each series, we keep the top K most related series, and set them as auxiliary series to assist target series forecasting. We compute ranks for each column of each matrix, to get the ranking matrix R_Γ. Through R_Γ, we select the top K series for each series.

4.2 ReIndex

The above training data $\mathcal{X}' = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(M\times S)}\}$ contains $M \times S$ samples in shape of $L \times (1 + K)$. For simplicity, we label the i-th sample as $\boldsymbol{x}^{(i)} \in \mathbb{N}^{L\times(1+K)}$. If we follow existing work to sample batches in the S dimension, that is, each batch will be allocated $b \times M \times (1 + K)$ data, and each with the length of L, as shown in the middle of Figure 4, where b denotes the batch size. According to existing transformer methods [42, 78], the average channel number is 200, that is $M = 200$. For such a relatively small number of channels, most of the DNNs including linear, CNNs, RNNs, and transformers should not have problems. However, in high-dimensional datasets, transformers may incur memory issues. In this work, to adapt transformer to high-dimension data, we design a more flexible method called ReIndex. Specifically, we sample the batch from the $M \times S$ dimensions, as shown on the right-hand side of Figure 4. With ReIndex, the number of samples of each batch can be reduced from $b \times M \times (1 + K)$ to $b \times (1 + K)$, each with a length of L. This indicates that the memory usage and computational complexity of the attention map are largely decreased, so we can set more flexible batch sizes. By setting shuffle for the training data, ReIndex improves the diversity of each training batch from specific windows in all M series to any window in any series. With more diversified training examples, ReIndex is also able to improve the model's generalization ability apart from the improved memory

[3] https://cython.org/

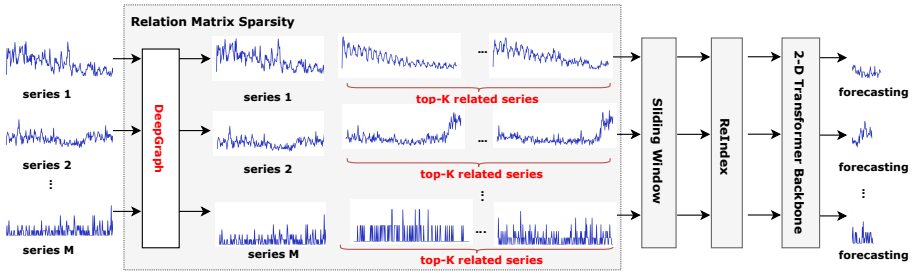


Figure 3: Overview of STHD, correlations of Multivariate Time Series data from different aspects are computed and then sparsed. That is, each series reserves the top K auxiliary series. Then, the target series together with K auxiliary series are input to the 2-D Transformer Backbone, which can extract the representation of 2-D input.

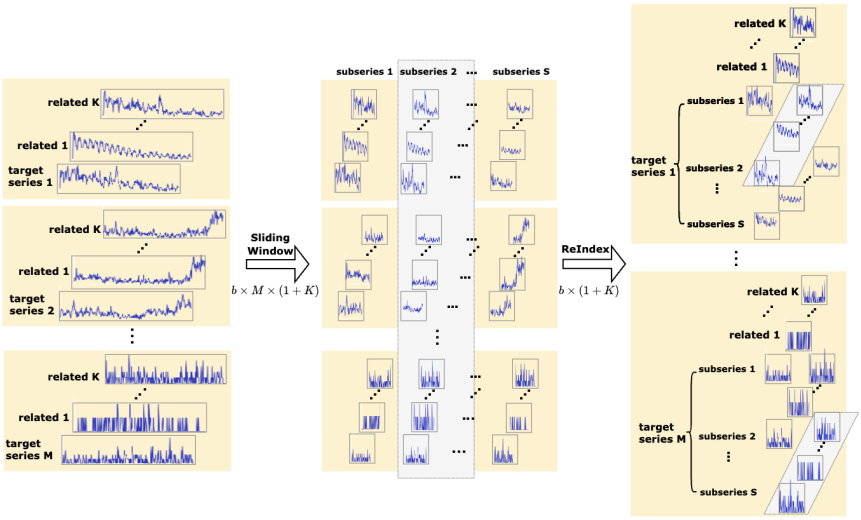


Figure 4: ReIndex process before batch sampling. Yellow square represents a series sample, including one target series, and K related series. All $M \times (1+K)$ series are split to $M \times (1+K) \times S$ subseries, and each of length L. Existing works sample batches on the S dimension. For example, the grey square in the middle of the figure denotes a batch of samples, with number of samples $b \times M \times (1+K)$, which largely increases memory usage. ReIndex reshapes the windows to $(M \times S) \times (1+K)$ and samples batches on the $M \times S$ dimension. The number of samples of each batch is $b \times (1+K)$.

usage and computational complexity in computing the attention map.

4.3 2-D Transformer

We adapt the vanilla Transformer to encode the 2-D input in the shape of $\mathbb{N}^{L\times(1+K)}$ as latent representations, where $(1+K)$ represents 1 target series and K auxiliary series.

Patches. Patches have been proven to be effective in previous works [42, 78] due to their ability to learn long and short-term temporal dependencies. Here for each series, first we pad l repeated numbers that equal the last value at the end of the original series. Then we divide each series $\boldsymbol{x}$ of length L into P patches

$$\boldsymbol{x}^{(i)} = \{\boldsymbol{x}_1^{(i)}, \boldsymbol{x}_2^{(i)}, ..., \boldsymbol{x}_P^{(i)}\} \in \mathbb{N}^{P\times l\times(1+K)} \tag{1}$$

where l is the length of a patch, $P = \left\lfloor \frac{L-l}{s} \right\rfloor + 2$ is the number of patches for each series, s represents strides of getting patches.

Encoder. First, we embed each patch $\boldsymbol{x}_p^{(i)} \in \mathbb{N}^{l\times(1+K)}$ into the D-dimension latent space of transformer by a trainable linear function $\mathbf{W}_p \in \mathbb{R}^{D\times l}$, and apply a sine and cosine position encoding for patches, $\mathbf{W}_{pos_{tem}}$, to distinguish the temporal order,

$$\boldsymbol{z}_p^{(i)} = \mathbf{W}_p \boldsymbol{x}_p^{(i)} + \mathbf{W}_{pos_{tem}}[:, :1+K] \tag{2}$$

Then each sample with all the patches $\boldsymbol{z}^{(i)} \in \mathbb{R}^{D\times(1+K)\times P}$. Next, we apply positional encoding for channels, $\mathbf{W}_{pos_{cha}}$, to distinguish the target series and auxiliary series, resulting in

$$\boldsymbol{z}_d^{(i)} = \boldsymbol{z}^{(i)} + \mathbf{W}_{pos_{cha}}[:, :1+K] \tag{3}$$

Final output $O^{(i)} = MultiHeadAtn\left(z_d^{(i)}, z_d^{(i)}, z_d^{(i)}\right) \in \mathbb{R}^{(P\times(1+K))\times D)}$. After adding a residual connection, followed by layer normalization $z_d^{(i)'} = LayerNorm_1\left(z_d^{(i)} + \ldots\right)$, $z_d^{(i)'}$ is passed through the feed-forward layer, comprising two 1-D convolutional layers $Conv_1()$ and $Conv_2()$.

$$z_d^{(i)'} = LayerNorm_2\left(z_d^{(i)} + Conv_2\left(GeLU\left(Conv_1\left(z_d^{(i)'}\right)\right)\right)\right) \quad (4)$$

where $GeLU()$ is the Gaussian Error Linear Unit active function.

Attention Mechanism. The attention mechanism is the most important element in each layer of the Transformer backbone. Existing works for exploring cross-time and channel relations are not enough. Channel-independent models like PatchTST use self-attention to explore time dependencies of different patches, thus ignoring the relation of channels; channel-dependent models like Crossformer use two-stage Attention, which explores relations separately. That is, they first explore time relations, then explore channel relations. Therefore, relations for patches from different series during different periods are ignored. In our work, we use self-attention to learn across the time and channels of target series and related series. Here we set attention head $h \in [1, H]$. For each head, we transform $z_d^{(i)}$ into query matrix, key matrix, and value matrix. Then we get the attention output $O_h^{(i)} \in \mathbb{R}^{(P\times(1+K))\times D''}$. To facilitate the integration of diverse information contexts captured by the distinct attention heads, the output of each individual attention head is concatenated into a unified representation, which is subsequently linearly transformed back to the original dimensional space D of Transformer.

$$O^{(i)} = Concat(O_1^{(i)}, O_2^{(i)}, \ldots, O_H^{(i)})\mathbf{W}_o \quad (5)$$

where *Concat()* denotes the concatenation operation, and $\mathbf{W}_o \in \mathbb{R}^{(H\times D'')\times D}$ represents the weight matrix of the subsequent linear transformation layer mapping the concatenated output back to the model's original dimensional space.

Decoder. The decoder transforms shape of $z_d^{(i)'}$ from $\mathbb{R}^{P\times(1+K)\times D}$ to $\mathbb{R}^{(1+K)\times D}$ that is suitable for the forecasting task. Specifically, the decoder first flattens the P dimension of $z_d^{(i)'}$, then applies a linear projection to it. Finally, we get the representation of (1+K) series. STHD gets the first vector as the representation of the target series, and then transforms it to $\hat{y^{(i)}} \in \mathbb{R}^{\tau}$ that equals the horizon.

Loss Function. Following common practice [42, 78], we use Mean Squared Error (MSE) and Adam optimizer to minimize the divergence of forecasting values and ground truth values. The loss in each series is gathered and averaged to get the overall loss $L = \frac{1}{M\times S}\sum_{i=1}^{M\times S}(y^{(i)} - \hat{y^{(i)}})^2$, where $y^{(i)}$ is the true value for the i_{th} sample, $\hat{y^{(i)}}$ is the predicted value.

5 Experiments

In this section, we report our extensive experiments on real-world high-dimensional MTS datasets to answer five questions: (1) How is the performance of our STHD approach in comparison with other counterparts for high-dimensional MTS forecasting tasks (for both effectiveness and memory issues)? (2) What is the impact of using related series? (3) How effective is the efficiency of DeepGraph? (4) How does the hyperparameter K influence the final performance? (5) How does ReIndex contribute to the forecasting performance? These questions can be matched to Section 5.3, Section 5.4, Section 5.5, Section 5.6, and Section 5.7 respectively.

5.1 Datasets

Three real-world high-dimensional MTS datasets are used for evaluation, namely Crime-Chicago, Wiki-People, and Traffic. Crime-Chicago and Wiki-People are larger in channel dimensions, the length is not very long. Traffic has relatively small channel dimensions, while it is longer than the other two datasets. Details are in Table 1.

Table 1: Details of multivariate datasets.

	Crime-Chicago	Wiki-People	Traffic
channels	1,155	6,107	862
length	260	550	17,545
frequency	monthly	daily	hourly

Crime-Chicago [4] extracted from the Chicago Police Department's Citizen Law Enforcement Analysis and Reporting System, reflects reported incidents of crime that occurred in the City of Chicago. We reserve the top 14 frequent crime types, aggregate other types of crime as one type across 77 communities, and process it as the monthly time series. *Wiki-People* [5] records Wikipedia web traffic and was published on Kaggle competition. It is daily time series data. When processing, first we filter out the 27,786 series with missing data and 9,194 series not published in Wikipedia agent. Then we filter articles that do not have series on all access, which results in 16,117 articles (93,020 series). Finally, we selected the 'people' topic, which resulted in 1,013 articles (6,107 series). *Traffic* [6] records the hourly time series of road occupancy rates measured by different sensors on San Francisco Bay area freeways.

5.2 Baselines and Experimental Settings

Baselines. We compare our STHD model with the following representative and state-of-the-art algorithms, including 1) channel-independent models: Linear, DLinear, Transformer, Non-stationary Transformer, PatchTST, TimesNet; 2) channel-dependent models: Crossformer, iTransformer; and 3) the Naïve method, the most basic forecasting baseline that the forecasting value equal to the past value. All the methods apart from Naïve are implemented on TSlib [7].

Parameter settings. All of models follow the same horizon length $\tau \in \{6, 12, 18, 24\}$ for Crime-Chicago, $\tau \in \{28, 35, 42, 49\}$ for Wiki-People, and $\tau \in \{96, 192, 336, 720\}$ for Traffic. We use long-term horizons on Traffic dataset to facilitate comparisons with existing studies, thereby simplifying the process of contrasting our findings with established research [42]. For the Traffic dataset, we follow most of the parameter settings as PatchTST, leaving the test batch size larger, at the same time, ensuring all the test samples are compared. Then we use grid-search to tune parameters, specifically, embedding size

[4] https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2
[5] https://www.kaggle.com/competitions/web-traffic-time-series-forecasting/
[6] https://pems.dot.ca.gov/
[7] https://github.com/thuml/Time-Series-Library

Table 2: Experimental results comparing with baselines. The horizon length values are from Section 5.2. The best results are in bold and the second best are underlined. To save space, NTransformer is the abbreviation of Non-stationary Transformer.

Dataset	Crime-Chicago															
	RMSE				WRMSPE				MAE				WAPE			
Horizon	6	12	18	24	6	12	18	24	6	12	18	24	6	12	18	24
Naive	1.334	1.340	1.440	1.371	1.088	1.081	1.157	1.104	0.827	0.816	0.901	0.865	0.674	0.659	0.724	0.697
Linear	2.006	1.794	1.729	1.685	1.635	1.448	1.389	1.357	1.586	1.358	1.303	1.263	1.293	1.096	1.047	1.017
DLinear	1.033	1.121	1.133	1.139	0.842	0.905	0.910	0.917	0.655	0.718	0.719	0.725	0.534	0.579	0.578	0.584
Transformer	1.369	1.385	1.384	1.371	1.116	1.118	1.112	1.104	0.886	0.892	0.882	0.879	0.722	0.720	0.709	0.708
NTransformer	1.099	1.148	1.166	1.176	0.896	0.927	0.936	0.947	0.678	0.703	0.715	0.722	0.553	0.567	0.574	0.581
Crossformer	1.177	1.278	1.286	1.293	0.959	1.032	1.033	1.041	0.750	0.825	0.822	0.826	0.611	0.666	0.661	0.665
TimesNet	1.076	1.120	1.152	1.157	0.877	0.910	0.925	0.932	0.665	0.691	0.706	0.710	0.542	0.558	0.567	0.572
PatchTST	1.041	1.122	1.117	1.087	0.848	0.904	0.897	0.877	0.640	0.685	0.678	0.660	0.522	0.551	0.545	0.533
iTransformer	1.030	1.096	1.131	1.142	0.839	0.885	0.909	0.920	0.632	0.665	0.685	0.696	0.515	0.537	0.551	0.561
STHD	**0.745**	**0.795**	**0.824**	**0.833**	**0.551**	**0.580**	**0.598**	**0.604**	**0.448**	**0.476**	**0.498**	**0.507**	**0.331**	**0.348**	**0.361**	**0.368**
Dataset	Wiki-People															
	RMSE				WRMSPE				MAE				WAPE			
Horizon	28	35	42	49	28	35	42	49	28	35	42	49	28	35	42	49
Naive	1.696	1.741	1.780	-	1.539	1.576	1.609	-	0.993	1.025	1.056	-	0.901	0.928	0.954	-
Linear	1.433	1.492	1.552	1.601	1.300	1.351	1.402	1.443	0.805	0.840	0.878	0.906	0.731	0.760	0.793	0.817
DLinear	1.373	1.434	1.500	1.569	1.246	1.298	1.355	1.414	0.767	0.805	0.840	0.880	0.696	0.729	0.759	0.793
Transformer	1.920	1.919	1.922	1.928	1.742	1.738	1.736	1.737	1.193	1.197	1.191	1.196	1.082	1.083	1.076	1.078
NTransformer	1.455	1.497	1.517	1.551	1.320	1.355	1.371	1.398	0.833	0.862	0.876	0.899	0.756	0.781	0.791	0.810
Crossformer	1.669	1.698	1.719	1.745	1.514	1.537	1.553	1.573	0.929	0.947	0.972	0.995	0.843	0.858	0.878	0.897
TimesNet	1.440	1.471	1.505	1.539	1.307	1.332	1.360	1.387	0.822	0.842	0.865	0.886	0.746	0.762	0.781	0.798
PatchTST	1.387	1.430	1.462	1.497	1.259	1.294	1.321	1.349	0.783	0.810	0.830	0.852	0.710	0.733	0.750	0.768
iTransformer	1.361	1.408	1.451	1.487	1.235	1.275	1.311	1.340	0.758	0.789	0.816	0.837	0.688	0.714	0.737	0.754
sthd	**1.188**	**1.231**	**1.272**	**1.307**	**1.141**	**1.180**	**1.216**	**1.246**	**0.658**	**0.684**	**0.708**	**0.732**	**0.632**	**0.655**	**0.676**	**0.698**
Dataset	Traffic															
	RMSE				WRMSPE				MAE				WAPE			
Horizon	96	192	336	720	96	192	336	720	96	192	336	720	96	192	336	720
Naive	1.199	0.968	0.807	-	1.500	1.209	1.005	-	0.562	0.417	0.311	-	0.704	0.521	0.387	-
Linear	0.650	0.669	0.673	0.698	0.813	0.835	0.839	0.867	0.301	0.320	0.320	0.341	0.377	0.400	0.398	0.424
DLinear	0.651	0.670	0.676	0.694	0.815	0.836	0.841	0.862	0.304	0.322	0.324	0.335	0.380	0.402	0.404	0.417
Transformer	0.875	0.889	0.862	0.855	1.096	1.110	1.074	1.062	0.431	0.446	0.421	0.411	0.539	0.556	0.524	0.511
NTransformer	0.809	0.813	0.838	0.849	1.012	1.015	1.044	1.054	0.381	0.382	0.410	0.412	0.477	0.477	0.510	0.512
Crossformer	0.826	0.864	0.869	-	1.033	1.079	1.082	-	0.400	0.432	0.431	-	0.501	0.540	0.536	-
TimesNet	0.776	0.786	0.794	0.808	0.971	0.981	0.989	1.004	0.329	0.338	0.346	0.346	0.411	0.422	0.431	0.430
PatchTST	0.647	0.657	0.668	0.649	0.809	0.820	0.832	0.854	0.313	0.316	0.324	**0.291**	0.392	0.394	0.404	**0.383**
iTransformer	0.723	0.686	0.666	0.722	0.904	0.856	0.830	0.897	0.370	0.323	0.298	0.351	0.463	0.403	0.371	0.436
STHD	**0.581**	**0.602**	**0.608**	**0.648**	**0.723**	**0.747**	**0.753**	**0.801**	**0.281**	**0.292**	**0.297**	0.319	**0.350**	**0.363**	**0.368**	0.394

'-' indicates the results couldn't be obtained due to the methods setting or GPU memory limitation. For the Naive method, if the input length is less than the output length, then it cannot be predicted. For GPU memory issues, please refer to Memory Discussion in Section 5.3.

for fully connected layers $\in \{64, 128, 256, 384\}$, embedding size for model $\in \{64, 128, 256, 384\}$, learning rate $\in \{0.1, 0.01, 0.001, 0.0001\}$, encoder layer, decoder layer and attention head $\in \{1, 2, 4, 6\}$ if used. Patch length $L = 12$ for Crime-Chicago, $L = 14$ for Wiki-People, and $L = 24$ for Traffic and stride of patches $s = 6$ for Crime-Chicago, $s = 7$ for Wiki-people, and $s = 12$ for Traffic. More details about baselines are listed in the Appendix.

Evaluation Metrics. Referring to [18], we select metrics covering different aspects: rooted mean squared measure without scaling: Rooted Mean Squared Error(RMSE), rooted mean squared measure scaling with a sum over test set: Weighted Root Mean Squared Percentage Error(WRMSPE), absolute measure without scaling: Mean Absolute Error(MAE), absolute measure scaling with a sum over test set: Weighted Absolute Percentage Error(WAPE). All metrics are the measurement of errors, which means they are the smaller the better.

5.3 Results Comparison

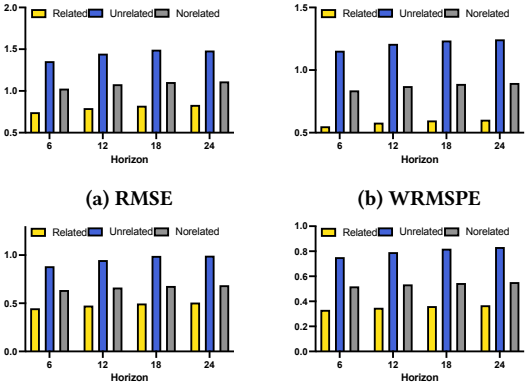
Table 2 presents the performance of all comparison methods on the three real-world datasets, where several interesting findings are observed as follows. 1) As the simplest baseline - Naïve, where the forecasting value equals its closest value. It is competitive with basic deep learning model: Linear and Transformer; 2) Linear-based methods, including Linear and DLinear, perform better than the traditional transformer-based models but worse than the patch-based transformer models in general; 3) Transformer-based channel-independent models, including Transformer, Non-stationary Transformer, TimesNet, and PatchTST demonstrate quite different performance on three datasets. PatchTST consistently performs better on three datasets, verifying the effectiveness of patches for capturing temporal dependencies on high-dimensional MTS data; 4) The transformer-based channel-dependent baselines, iTransformer performs better than Crossformer, even overall better than channel-independent models on Wiki-People. Crossformer shows a quite stable performance that can be ranked at the middle of all baselines on high-dimensional MTS data. However, during the evaluation process, it is very time-consuming to run both, making it hard to scale well on high-dimension MTS data due to its modeling of full relations of all channels; 5) Finally, our STHD earns an overall best performance on all three high-dimensional datasets, with improvements of 30.02%, 10.20%, and 5.43% on Crime-Chicago, Wiki-People, and Traffic respectively.

Memory Discussion. Training with a single A100 graphic card, CUDA out-of-memory error occurs when we use Crossformer at 720

horizons on the Traffic dataset. Since Traffic has a large S, samples of each batch are allocated M times more than using ReIndex, which costs more parameters.

5.4 Related Series

To better demonstrate the impact of the related series, we made the following comparison: a) target series with top K related series as the input; b) target series with top K unrelated series as input to our method STHD; c) target series without any series. Figure 5 shows the result on Crime-Chicago. Yellow, blue, and grey bars represent experimental results from a), b), and c) separately. With related series, yellow bars are overall twice as good as the blue bars across all measures. Without related series, grey bars are overall better than blue bars, which demonstrates the importance of reducing the introduction of noise from unrelated series. To sum up, the related series significantly benefits the task, demonstrated by its improved performance on different measures.



(a) RMSE (b) WRMSPE

(c) MAE (d) WAPE

Figure 5: Effect of Related Series on Crime-Chicago. Yellow, blue and grey bars represent the target series with K related series, with K unrelated series, and without auxiliary series.

5.5 Efficiency of DeepGraph

To verify the efficiency of DeepGraph, we compare the run time of computing correlations with the DeepGraph graph structure and computing without DeepGraph (i.e., We use the correlation method embedded in Nerd instead of DeepGraph). We compare them to the Wiki-People dataset considering its largest number of dimensions compared with the other two datasets.

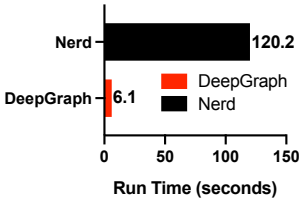
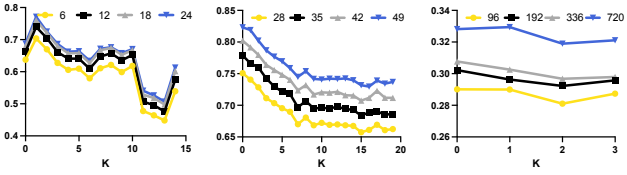


Figure 6: Comparison of the Run Time for Correlation Computing.

Running time is shown in Figure 6, we can observe that Nerd in black bars is 120.2 seconds, while DeepGraph in red bars are 6.1 seconds. In this way, we achieve a 20 times speedup compared with a naïve pair-wise computation.

5.6 Correlation Sparsity

Figure 7 shows the experimental results for the change of hyper-parameter K on Crime-Chicago, Wiki-People, and Traffic, respectively. A similar trend with the change of K on all the datasets is



(a) Crime-Chicago (b) Wiki-People (c) Traffic

Figure 7: MAE measure with the change of K on three datasets. The x-axis is the value of K, the y-axis is MAE, and the line in yellow, black, grey, and blue represents the horizon in 6,12,18,24 separately.

observed. With the increase of K, the performance first obviously drop, and then slightly rise. The possible reason is that at the very beginning, the increased K means the model can benefit from real related series, while when K becomes too large, the series becomes less relevant which will introduce noise to the model.

5.7 ReIndex

Note that ReIndex is designed to alleviate the memory issue in computing the attention map and it makes the training stage scalable to larger batch sizes, as shown in Figure 4 and Appendix. However, we find that apart from this, ReIndex is also able to improve the model's effectiveness as shown in Table 3. In this table, the following comparison is conducted on the Crime-Chicago dataset: take the target series as the input, and use STHD with ReIndex $STHD_{wt}$ and without ReIndex $STHD_{wto}$ to forecast. The results show that $STHD_{wt}$ outperforms $STHD_{wto}$ across all measures across all horizons.

Table 3: Experimental results for ReIndex. $STHD_{wt}$ and $STHD_{wto}$ represent the extensive experiment that takes target series as input for STHD with ReIndex and without ReIndex respectively. Best results are shown with underline.

	RMSE			WRMSPE			WAPE		
Horizon	6	12	18	6	12	18	6	12	18
$STHD_{wt}$	1.028	1.081	1.107	0.838	0.872	0.889	0.519	0.535	0.545
$STHD_{wto}$	1.041	1.122	1.117	0.849	0.904	0.898	0.522	0.551	0.545

6 Conclusion

This paper proposed a scalable Transformer framework - STHD, designed specifically for tackling the intricate challenges of high-dimensional Multivariate Time Series (MTS) forecasting. Through the innovative incorporation of sparse correlation matrix, STHD significantly enhances forecasting performance by efficiently filtering out unrelated series. Furthermore, the introduction of the ReIndex strategy emerges as a dual-purpose solution, mitigating the constraints of parameter ranges in training transformer-based methods on high-dimensional MTS data, while at the same time enriching

the diversity of shuffled training samples. Experimental results underscore STHD's forecasting performance in high-dimensional MTS data. In essence, STHD model focuses on the correlation of high-dimensional data with accurate forecasts. Further, we will continue to refine and expand upon the STHD model, with modalities and will further explore the co-evaluation of high-dimensional MTS data.

Appendix

A Data Analysis and Processing

Before this work, none of the time series works used the Crime-Chicago dataset for MTS forecasting, so we conducted data analysis for Crime-Chicago. Figure 8 shows the correlation of different communities, the x-axis, and y-axis represent the community area. To draw this figure, we add up all crime types of the series from the same community. The color scale on the right indicates the strength and direction of the correlation coefficient: Dark blue represents a correlation coefficient close to 1, indicating a strong positive correlation; Lighter blue and white represent correlation coefficients closer to 0, indicating little or no linear correlation; The red shades (which are not prominently visible in this matrix) would indicate a negative correlation, with dark red being close to -1. Through the figure, we can clearly see different color distributions, which illustrates why STHD uses the correlation method. For more details about analyzing and pre-processing, please refer to Colab [8].

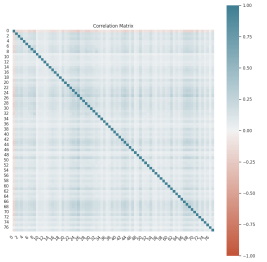


Figure 8: The correlation of communities on Crime-Chicago. The color scale on the right indicates the strength and direction of the correlation coefficient.

B Theoretical Analysis

Time. Relation Matrix Sparsity module, with its $O(M^2)$ complexity of M series, is the most computationally intensive part, primarily due to pairwise correlation computations. Because of that, we leverage parallel computing and efficient data structures to mitigate the computational load. The ReIndex module, while crucial for data optimization, earns the same complexity as other works. The Transformer-based model's primary complexity stems from its attention mechanism. Unlike PatchTST, which focuses solely on time dependency with a complexity of $O(P^2)$ with P patches within a subseries, Adapted Transformer's complexity is primarily dependent on the attention: $O(((1+K)SP)^2)$ for S subseries of each series. However, previous Sparsity and Reindex steps effectively reduce the input size where $K < 20, (1+K) << M$, mitigating the quadratic dependency. Improvements in accuracy for high-dimensional datasets also counterbalance this.

Memory. Relation Matrix Sparsity uses $O(MK)$ where $K << M$ is the most significant correlation. For M time series, the naive method stores a full correlation matrix that requires $O(M^2)$. ReIndex reduces samples from $b \times M \times (1+K)$ to $b \times (1+K)$. Adapted Transformer: needs extra memory than other transformers for attention $O((KS)^2)$ and related layers.

[8] https://colab.research.google.com/drive/1u9utf8ZadWTWo73s_l4Uh6dETLvleNts?usp=sharing

C Experimental Setting

All DNNs models including pre-test, ablation study on STHD, and baselines are implemented in PyTorch and trained on a single NVIDIA A100 GPU. The parameter configurations for the Crime-Chicago and Wiki-People are outlined in Table 4. For Traffic, we follow the setting of PatchTST. To allow each model to converge during training, we set the training epoch to 100, and use an early stop setting, which means if the delta of the validate set loss is less than e-7, the training will stop and begin to test.

Table 4: Parameters in Crime-Chicago and Wiki-People.

Dataset	Chicago-Crime					
	d_{ff}	d_{model}	e_{layer}	lr	b	n_{head}
Linear	-	128	-	0.01	128	-
DLinear	-	384	-	0.0001	128	-
Transformer	384	256	2	0.001	32	4
NTransformer	384	256	1	0.001	32	4
Crossformer	384	256	2	0.01	32	4
PatchTST	384	256	2	0.001	32	4
TimesNet	384	256	2	0.001	32	4
iTransformer	384	256	2	0.001	32	4
STHD	384	256	2	0.001	128	4
Dataset	Wiki-People					
Linear	-	128	-	0.01	16	-
DLinear	-	128	-	0.01	32	-
Transformer	128	128	1	0.001	32	4
NTransformer	128	128	1	0.001	32	4
Crossformer	128	128	2	0.001	32	4
PatchTST	128	128	1	0.001	32	4
TimesNet	128	128	2	0.001	32	4
iTransformer	128	128	1	0.001	32	4
STHD	128	128	1	0.001	128	4

D Performance on Low-dim Datasets

While the STHD is designed with high-dimensional MTS forecasting, theoretically, it can be applied to low-dimensional series. Some further experiments on lower-dimensional data have been done on Weather. Results are listed in Table 5.

Table 5: Experimental results on low-dimensional dataset. The best results are in bold and the second best underlined.

	MAE				RMSE			
Horizon	96	192	336	720	96	192	336	720
DLinear	0.243	0.422	0.281	0.468	0.320	0.515	0.372	0.574
Crossformer	0.214	0.384	0.265	0.442	0.335	0.517	0.416	0.605
PatchTST	0.205	0.392	0.245	0.444	0.284	0.498	0.335	0.567
TimesNet	0.274	0.399	0.287	0.505	0.287	**0.419**	**0.333**	**0.482**
iTransformer	0.211	0.401	0.251	0.453	0.290	0.506	0.338	0.571
STHD	**0.187**	**0.316**	**0.233**	**0.388**	**0.276**	0.452	0.331	0.530

We have the following findings: 1) channel-dependent model Crossformer performs well when the horizon equals 192 and 720 on MAE, which may contribute to less noise introduced in low-dimensional datasets; 2) patch-based method PatchTST and multi-frequency-based method TimesNets show their ability to capture long-term dependencies; 3) by sampling correlated series ($K = 1$), STHD still works on low-dimensional datasets.

References

[1] Han Bao, Xun Zhou, Yiqun Xie, Yanhua Li, and Xiaowei Jia. 2022. STORM-GAN: Spatio-Temporal Meta-GAN for Cross-City Estimation of Human Mobility Responses to COVID-19. In Proceedings of IEEE International Conference on Data Mining (ICDM).

[2] Shaik Johny Basha, Tamminina Ammannamma, Kolla Vivek, and Venkata Srinivasu Veesam. 2022. Comparative Analysis of Time Series Forecasting Models to Predict Amount of Rainfall in Telangana. In Proceedings of International Conference on Advanced Computing and Communication Systems (ICACCS).

[3] George E.P. Box and Gwilym M. Jenkins. 1970. Time Series Analysis: Forecasting and Control. Holden-Day.

[4] Zekun Cai, Renhe Jiang, Xinyu Yang, Zhaonan Wang, Diansheng Guo, Hill Hiroki Kobayashi, Xuan Song, and Ryosuke Shibasaki. 2023. MemDA: Forecasting Urban Time Series with Memory-based Drift Adaptation. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM).

[5] Ching Chang, Chiao-Tung Chan, Wei-Yao Wang, Wen-Chih Peng, and Tien-Fu Chen. 2024. TimeDRL: Disentangled Representation Learning for Multivariate Time-Series. In Proceedings of IEEE International Conference on Data Engineering (ICDE).

[6] Ling Chen, Donghui Chen, Zongjiang Shang, Binqing Wu, Cen Zheng, Bo Wen, and Wei Zhang. 2024. Multi-Scale Adaptive Graph Neural Network for Multivariate Time Series Forecasting. In Proceedings of IEEE International Conference on Data Engineering (ICDE).

[7] Shengyu Chen, Nasrin Kalanat, Simon Topp, Jeffrey Sadler, Yiqun Xie, Zhe Jiang, and Xiaowei Jia. 2023. Meta-Transfer-Learning for Time Series Data with Extreme Events: An Application to Water Temperature Prediction. In Proceedings of ACM International Conference on Information and Knowledge Management(CIKM).

[8] Yuqi Chen, Hanyuan Zhang, Weiwei Sun, and Baihua Zheng. 2023. RNTrajRec: Road Network Enhanced Trajectory Recovery with Spatial-Temporal Transformer. In Proceedings of IEEE International Conference on Data Engineering (ICDE).

[9] Yue Cui, Kai Zheng, Dingshan Cui, Jiandong Xie, Liwei Deng, Feiteng Huang, and Xiaofang Zhou. 2021. METRO: a generic graph neural network framework for multivariate time series forecasting. Proc. VLDB Endow. 15, 2 (2021).

[10] Wenying Duan, Xiaoxi He, Zimu Zhou, Lothar Thiele, and Hong Rao. 2023. Localised Adaptive Spatial-Temporal Graph Neural Network. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD). 448–458.

[11] Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD).

[12] Yujie Fan, Chin-Chia Michael Yeh, Huiyuan Chen, Yan Zheng, Liang Wang, Junpeng Wang, Xin Dai, Zhongfang Zhuang, and Wei Zhang. 2023. Spatial-Temporal Graph Boosting Networks: Enhancing Spatial-Temporal Graph Neural Networks via Gradient Boosting. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM). Association for Computing Machinery.

[13] Yang Fan Chiang, Chun-Wei Shen, Jhe-Wei Tsai, Pei-Xuan Li, Tzu-Chang Lee, and Hsun-Ping Hsieh. 2023. ParkFlow: Intelligent Dispersal for Mitigating Parking Shortages Using Multi-Granular Spatial-Temporal Analysis. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM). Association for Computing Machinery.

[14] Aosong Feng and Leandros Tassiulas. 2022. Adaptive Graph Spatial-Temporal Transformer Network for Traffic Forecasting. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM). Association for Computing Machinery.

[15] LaLao Gao, DingJun Zhang, MingChao Liao, and ZhiQiang Huang. 2022. Multivariate time series prediction based on graph convolutional neural networks. In Proceedings of International Conference on Artificial Intelligence, Information Processing and Cloud Computing (AIIPCC).

[16] Sebastian Gerard, Yu Zhao, and Josephine Sullivan. 2023. WildfireSpreadTS: A dataset of multi-modal time series for wildfire spread prediction. In Proceedings of Conference on Neural Information Processing Systems Datasets and Benchmarks Track (NIPS).

[17] Shengnan Guo, Youfang Lin, Letian Gong, Chenyu Wang, Zeyu Zhou, Zekai Shen, Yiheng Huang, and Huaiyu Wan. 2023. Self-Supervised Spatial-Temporal Bottleneck Attentive Network for Efficient Long-term Traffic Forecasting. In Proceedings of IEEE International Conference on Data Engineering (ICDE).

[18] Hansika Hewamalage, Klaus Ackermann, and Christoph Bergmeir. 2023. Forecast evaluation for data scientists: common pitfalls and best practices. Data Mining and Knowledge Discovery 37, 2 (2023).

[19] Mingzhi Hu, Zhuoyun Zhong, Xin Zhang, Yanhua Li, Yiqun Xie, Xiaowei Jia, Xun Zhou, and Jun Luo. 2023. Self-supervised Pre-training for Robust and Generic Spatial-Temporal Representations. In Proceedings of IEEE International Conference on Data Mining (ICDM).

[20] Jizhou Huang, Zhengjie Huang, Xiaomin Fang, Shikun Feng, Xuyi Chen, Jiaxiang Liu, Haitao Yuan, and Haifeng Wang. 2022. DuETA: Traffic Congestion Propagation Pattern Modeling via Efficient Graph Learning for ETA Prediction at Baidu Maps. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM). Association for Computing Machinery.

[21] Ping-Chia Huang, Yueh-Li Chen, Yi-Syuan Liou, Bing-Chen Tsai, Chun-Chieh Wu, and Winston H. Hsu. 2023. STAMINA (Spatial-Temporal Aligned Meteorological INformation Attention) and FPL (Focal Precip Loss): Advancements in Precipitation Nowcasting for Heavy Rainfall Events. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM). Association for Computing Machinery.

[22] Rob J Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. 2008. Forecasting with Exponential Smoothing: The State Space Approach. Springer.

[23] Furong Jia, Kevin Wang, Yixiang Zheng, Defu Cao, and Yan Liu. 2024. GPT4MTS: Prompt-based Large Language Model for Multimodal Time-series Forecasting. Proceedings of the AAAI Conference on Artificial Intelligence (AAAI) (2024).

[24] Song Jiang, Tahin Syed, Xuan Zhu, Joshua Levy, Boris Aronchik, and Yizhou Sun. 2022. Bridging Self-Attention and Time Series Decomposition for Periodic Forecasting. In Proceedings of ACM International Conference on Information & Knowledge Management (CIKM).

[25] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. 2024. Time-LLM: Time series forecasting by reprogramming large language models. In Proceedings of International Conference on Learning Representations (ICLR).

[26] Abdelouahab Khelifati, Mourad Khayati, Anton Dignös, Djellel Difallah, and Philippe Cudré-Mauroux. 2023. TSM-Bench: Benchmarking Time Series Database Systems for Monitoring Applications. Proc. VLDB Endow. 16, 11 (2023).

[27] Zhichen Lai, Dalin Zhang, Huan Li, Christian S. Jensen, Hua Lu, and Yan Zhao. 2023. LightCTS: A Lightweight Framework for Correlated Time Series Forecasting. Proc. ACM Manag. Data 1, 2 (2023).

[28] Xiang Lan, Hanshu Yan, Shenda Hong, and Mengling Feng. 2024. Towards Better Time Series Contrastive Learning: A Dynamic Bad Pair Mining Approach. In Proceedings of the International Conference on Learning Representations (ICLR).

[29] Jongsoo Lee, Byeongtae Park, and Dong-Kyu Chae. 2023. DuoGAT: Dual Time-oriented Graph Attention Networks for Accurate, Efficient and Explainable Anomaly Detection on Time-series. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM).

[30] Jun Li and Ge Zhang. 2023. Fragment and Integrate Network (FIN): A Novel Spatial-Temporal Modeling Based on Long Sequential Behavior for Online Food Ordering Click-Through Rate Prediction. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM). Association for Computing Machinery.

[31] Yan Li, Xinjiang Lu, Haoyi Xiong, Jian Tang, Jiantao Su, Bo Jin, and Dejing Dou. 2023. Towards Long-Term Time-Series Forecasting: Feature, Pattern, and Distribution. In Proceedings of IEEE International Conference on Data Engineering (ICDE).

[32] Yile Liang, Donghui Li, Jiuxia Zhao, Xuetao Ding, Huanjia Lian, Jinghua Hao, and Renqing He. 2023. Enhancing Dynamic On-demand Food Order Dispatching via Future-informed and Spatial-temporal Extended Decisions. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM). Association for Computing Machinery.

[33] Shaohua Liu, Yu Qi, Gen Li, Mingjian Chen, Teng Zhang, Jia Cheng, and Jun Lei. 2023. STGIN: Spatial-Temporal Graph Interaction Network for Large-scale POI Recommendation. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM). Association for Computing Machinery.

[34] Shu Liu, Jiaheng Wang, Jiamin Chen, Jianliang Gao, and Yuhui Zhong. 2023. TemDep: Temporal Dependency Priority for Multivariate Time Series Prediction. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM).

[35] Xiangyue Liu, Xinqi Lyu, Xiangchi Zhang, Jianliang Gao, and Jiamin Chen. 2022. Memory Augmented Graph Learning Networks for Multivariate Time Series Forecasting. In Proceedings of ACM International Conference on Information & Knowledge Management (CIKM).

[36] Xinyu Liu, Zijing Wei, Wenqing Yu, Shaozhi Liu, Gang Wang, Xiaoguang Liu, and Yusen Li. 2023. Khronos: A Real-Time Indexing Framework for Time Series Databases on Large-Scale Performance Monitoring Systems. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM).

[37] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In Proceedings of International Conference on Learning Representations (ICLR).

[38] Helmut Lütkepohl. 1985. Forecasting Contemporaneously Aggregated Vector ARMA Processes. Journal of Business & Economic Statistics 3, 4 (1985), 401–407.

[39] Minbo Ma, Jilin Hu, Christian S. Jensen, Fei Teng, Peng Han, Zhiqiang Xu, and Tianrui Li. 2024. Learning Time-aware Graph Structures for Spatially Correlated

Time Series Forecasting. In Proceedings of IEEE International Conference on Data Engineering (ICDE).
[40] Evangelos Spiliotis Mohsen Hamoudia, Spyros Makridakis. 2023. Forecasting with Artificial Intelligence: Theory and Applications. Palgrave Macmillan.
[41] Andrea Nascetti, RITU YADAV, Kirill Brodt, Qixun Qu, Hongwei Fan, Yuri Shendryk, Isha Shah, and Christine Chung. 2023. BioMassters: A Benchmark Dataset for Forest Biomass Estimation using Multi-modal Satellite Time-series. In Proceedings of Conference on Neural Information Processing Systems Datasets and Benchmarks Track (NIPS).
[42] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In Proceedings of International Conference on Learning Representations (ICLR).
[43] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. 2019. High-dimensional multivariate forecasting with low-rank Gaussian Copula Processes. In Proceedings of Advances in Neural Information Processing Systems (NIPS).
[44] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. 2019. Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting. In Advances in Neural Information Processing Systems (NIPS).
[45] Zezhi Shao, Zhao Zhang, Fei Wang, Wei Wei, and Yongjun Xu. 2022. Spatial-Temporal Identity: A Simple yet Effective Baseline for Multivariate Time Series Forecasting. In Proceedings of ACM International Conference on Information & Knowledge Management(CIKM).
[46] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. 2022. Pre-training Enhanced Spatial-temporal Graph Neural Network for Multivariate Time Series Forecasting. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD).
[47] Christopher A. Sims. 1980. Macroeconomics and Reality. Econometrica 48 (1980).
[48] Junho Song, Jiwon Son, Dong-hyuk Seo, Kyungsik Han, Namhyuk Kim, and Sang-Wook Kim. 2022. ST-GAT: A Spatio-Temporal Graph Attention Network for Accurate Traffic Speed Prediction. In Proceedings of ACM International Conference on Information & Knowledge Management (CIKM).
[49] Abishek Sriramulu, Nicolas Fourrier, and Christoph Bergmeir. 2023. Adaptive dependency learning graph neural networks. Information Sciences 625 (2023), 700–714.
[50] Chang Wei Tan, Christoph Bergmeir, Francois Petitjean, and Geoffrey I. Webb. 2020. Monash University, UEA, UCR Time Series Extrinsic Regression Archive. http://arxiv.org/abs/2006.10996
[51] Yihong Tang, Ao Qu, Andy H.F. Chow, William H.K. Lam, S.C. Wong, and Wei Ma. 2022. Domain Adversarial Spatial-Temporal Network: A Transferable Framework for Short-term Traffic Forecasting across Cities. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM). Association for Computing Machinery.
[52] Wanjie Tao, Zhang-Hua Fu, Liangyue Li, Zulong Chen, Hong Wen, Yuanyuan Liu, Qijie Shen, and Peilin Chen. 2022. A Dual Channel Intent Evolution Network for Predicting Period-Aware Travel Intentions at Fliggy. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM). Association for Computing Machinery.
[53] Dominik Traxl, Niklas Boers, and Jürgen Kurths. 2016. Deep graphs—A general framework to represent and analyze heterogeneous complex systems across scales. Chaos: An Interdisciplinary Journal of Nonlinear Science 26, 6 (2016).
[54] Chenyu Wang, Zongyu Lin, Xiaochen Yang, Jiao Sun, Mingxuan Yue, and Cyrus Shahabi. 2022. HAGEN: Homophily-Aware Graph Convolutional Recurrent Network for Crime Forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI).
[55] Dingsu Wang, Yuchen Yan, Ruizhong Qiu, Yada Zhu, Kaiyu Guan, Andrew Margenot, and Hanghang Tong. 2023. Networked Time Series Imputation via Position-aware Graph Enhanced Variational Autoencoders. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD). Association for Computing Machinery.
[56] Hao Wang and Zhenguo Zhang. 2022. TATCN: Time Series Prediction Model Based on Time Attention Mechanism and TCN. In Proceedings of IEEE International Conference on Computer Communication and Artificial Intelligence (CCAI).
[57] Jingyuan Wang, Chen Yang, Xiaohan Jiang, and Junjie Wu. 2023. WHEN: A Wavelet-DTW Hybrid Attention Network for Heterogeneous Time Series Analysis. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD).
[58] Min Wang, Hua Wang, and Fan Zhang. 2023. FAMC-Net: Frequency Domain Parity Correction Attention and Multi-Scale Dilated Convolution for Time Series Forecasting. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM).
[59] Yihe Wang, Yu Han, Haishuai Wang, and Xiang Zhang. 2023. Contrast Everything: A Hierarchical Contrastive Framework for Medical Time-Series. In Proceedings of Conference on Neural Information Processing Systems (NIPS).
[60] Yuan Wang, Zezhi Shao, Tao Sun, Chengqing Yu, Yongjun Xu, and Fei Wang. 2023. Clustering-property Matters: A Cluster-aware Network for Large Scale Multivariate Time Series Forecasting. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM).
[61] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. Timesnet: Temporal 2D-Variation Modeling for General Time Series Analysis. In Proceedings of the International Conference on Learning Representations (ICLR).
[62] Xinle Wu, Dalin Zhang, Miao Zhang, Chenjuan Guo, Bin Yang, and Christian S. Jensen. 2023. AutoCTS+: Joint Neural Architecture and Hyperparameter Search for Correlated Time Series Forecasting. Proc. ACM Manag. Data 1, 1 (2023).
[63] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD).
[64] Sheng Xiang, Dawei Cheng, Chencheng Shang, Ying Zhang, and Yuqi Liang. 2022. Temporal and Heterogeneous Graph Neural Network for Financial Time Series Prediction. In Proceedings of ACM International Conference on Information & Knowledge Management (CIKM).
[65] Jiandong Xie, Yue Cui, Feiteng Huang, Chao Liu, and Kai Zheng. 2022. MARINA: An MLP-Attention Model for Multivariate Time-Series Analysis. In Proceedings of ACM International Conference on Information & Knowledge Management (CIKM).
[66] Dongkuan Xu, Wei Cheng, Bo Zong, Dongjin Song, Jingchao Ni, Wenchao Yu, Yanchi Liu, Haifeng Chen, and Xiang Zhang. 2020. ensorized LSTM with Adaptive Shared Memory for Learning Trends in Multivariate Time Series. (2020).
[67] Jingwen Xu, Fei Lyu, and Pong C. Yuen. 2023. Density-Aware Temporal Attentive Step-wise Diffusion Model For Medical Time Series Imputation. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM).
[68] Junchen Ye, Zihan Liu, Bowen Du, Leilei Sun, Weimiao Li, Yanjie Fu, and Hui Xiong. 2022. Learning the Evolutionary and Multi-scale Graph Structure for Multivariate Time Series Forecasting. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD).
[69] Jinhui Yi, Huan Yan, Haotian Wang, Jian Yuan, and Yong Li. 2023. DeepSTA: A Spatial-Temporal Attention Network for Logistics Delivery Timely Rate Prediction in Anomaly Conditions. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM).
[70] Kun Yi, Qi Zhang, Wei Fan, Hui He, Liang Hu, Pengyang Wang, Ning An, Longbing Cao, and Zhendong Niu. 2023. FourierGNN: Rethinking Multivariate Time Series Forecasting from a Pure Graph Perspective. In Proceedings of Conference on Neural Information Processing Systems (NIPS).
[71] Chengqing Yu, Fei Wang, Zezhi Shao, Tao Sun, Lin Wu, and Yongjun Xu. 2023. DSformer: A Double Sampling Transformer for Multivariate Time Series Long-term Prediction. In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM).
[72] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are Transformers Effective for Time Series Forecasting? Proceedings of the AAAI Conference on Artificial Intelligence (AAAI) 37, 9 (2023).
[73] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A Transformer-based Framework for Multivariate Time Series Representation Learning. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD).
[74] Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. 2022. TFAD: A Decomposition Time Series Anomaly Detection Architecture with Time-Frequency Analysis. In Proceedings of ACM International Conference on Information & Knowledge Management (CIKM).
[75] Jiawen Zhang, Shun Zheng, Wei Cao, Jiang Bian, and Jia Li. 2023. Warpformer: A Multi-scale Modeling Approach for Irregular Clinical Time Series. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD).
[76] Jiawen Zhang, Shun Zheng, Wei Cao, Jiang Bian, and Jia Li. 2023. Warpformer: A Multi-scale Modeling Approach for Irregular Clinical Time Series. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD).
[77] Weijia Zhang, Hao Liu, Jindong Han, Yong Ge, and Hui Xiong. 2022. Multi-Agent Graph Convolutional Reinforcement Learning for Dynamic Electric Vehicle Charging Pricing. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD).
[78] Yunhao Zhang and Junchi Yan. 2023. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. In Proceedings of International Conference on Learning Representations (ICLR).
[79] Zhiqiang Zhang, Yuxuan Chen, Dandan Zhang, Yining Qian, and Hongbing Wang. 2023. CTFNet: Long-Sequence Time-Series Forecasting Based on Convolution and Time–Frequency Analysis. IEEE Transactions on Neural Networks and Learning Systems (2023), 1–15.
[80] Xiangyu Zhao, Wenqi Fan, Hui Liu, and Jiliang Tang. 2022. Multi-Type Urban Crime Prediction. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI).
[81] Xiaohui Zhao, Shuai Wang, Hai Wang, Tian He, Desheng Zhang, and Guang Wang. 2023. HST-GT: Heterogeneous Spatial-Temporal Graph Transformer for Delivery Time Estimation in Warehouse-Distribution Integration E-Commerce. In Proceedings of ACM International Conference on Information and Knowledge

Management (CIKM). Association for Computing Machinery.
[82] Yanjun Zhao, Ziqing‘ Ma, Tian Zhou, Mengni Ye, Liang Sun, and Yi Qian. 2023. GCformer: An Efficient Solution for Accurate and Scalable Long-Term Multivariate Time Series Forecasting. In Proceedings of ACM International Conference on Information and Knowledge Management(CIKM).
[83] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI).
[84] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In Proceedings of International Conference on Machine Learning (ICML).
[85] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. 2023. One Fits All: Power General Time Series Analysis by Pretrained LM. In Proceedings of Neural Information Processing Systems (NIPS).