ST-Norm: Spatial and Temporal Normalization for Multi-variate Time Series Forecasting

Jinliang Deng*
Australian Artificial Intelligence Institute, University of Technology Sydney
Sydney, Australia
jinliang.deng@student.uts.edu.au

Xiusi Chen
University of California, Los Angeles
Los Angeles, USA
xchen@cs.ucla.edu

Renhe Jiang
University of Tokyo, Center for Spatial Information Science
Tokyo, Japan
jiangrh@csis.u-tokyo.ac.jp

Xuan Song†
SUSTech-UTokyo Joint Research Center on Super Smart City, Department of Computer Science and Engineering, Southern University of Science and Technology (SUSTech)
Shenzhen, China
songx@sustech.edu.cn

Ivor W. Tsang†
Australian Artificial Intelligence Institute, University of Technology Sydney
Sydney, Australia
Ivor.Tsang@uts.edu.au

ABSTRACT

Multi-variate time series (MTS) data is a ubiquitous class of data abstraction in the real world. Any instance of MTS is generated from a hybrid dynamical system with their specific dynamics normally unknown. The hybrid nature of such a dynamical system is a result of complex external impacts, which can be summarized as high-frequency and low-frequency from the temporal view, or global and local if we take the spatial view. These impacts also determine the forthcoming development of MTS making them paramount to capture in a time series forecasting task. However, conventional methods face intrinsic difficulties in disentangling the components yielded by each kind of impact from the raw data. To this end, we propose two kinds of normalization modules – temporal and spatial normalization – which separately refine the high-frequency component and the local component underlying the raw data. Moreover, both modules can be readily integrated into canonical deep learning architectures such as Wavenet and Transformer. Extensive experiments[1] on three datasets are conducted to illustrate that, with additional normalization modules, the performance of the canonical architectures can be enhanced by a large margin in the application of MTS and achieves state-of-the-art results compared with existing MTS models.

*Also with Department of Computer Science and Engineering, Southern University of Science and Technology.
†Corresponding author
[1]Code available at: https://github.com/JLDeng/ST-Norm.git

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467330

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Time Series Forecasting, Deep Learning, Normalization

ACM Reference Format:
Jinliang Deng, Xiusi Chen, Renhe Jiang, Xuan Song, and Ivor W. Tsang. 2021. ST-Norm: Spatial and Temporal Normalization for Multi-variate Time Series Forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467330

1 INTRODUCTION

Time series forecasting is an imperative problem in many industrial and business applications. For instance, a public transport operator can allocate sufficient capacity to mitigate the queuing time in a region in advance, if they have the means to foresee that a particular region will suffer from a supply shortage in the next couple of hours [7]. Taking another example, an investor can avoid economic loss with the assistance of a robo-advisor which is able to predict a potential market crash [5]. Due to the complex and continuously fluctuation in impacting factors, real-world time series tends to be extraordinarily non-stationary, that is, exhibiting diverse dynamics. For instance, the traffic volume over a road is largely affected by the road's condition, location, and the current time and weather condition. In the case of retailing, the current season, price and brand serve as determinants for the sales of a merchandise. The diverse dynamics impose an enormous challenge on time series forecasting. In this work, we will study multi-variate time series forecasting, where multiple variables evolve with time.

Traditional time series forecasting algorithms, such as ARIMA and state space models (SSMs), provide a principled framework for modeling and learning time series patterns. However, these

algorithms have a rigorous requirement for the stationarity of a time series, which encounters severe limitation in practical use if most of the impacting factors are unavailable. With the recent advance on deep learning techniques, we are now capable of handling the complex dynamics as a single unit, even without any additional supplement of impacting factors. Common neural architectures applied on time series data include recurrent neural network (RNN), long-short term memory (LSTM) [8], Transformer [12], Wavenet [23] and temporal convolution networks (TCN) [2].

1.1 Preliminary Analysis

There is a rich quantity of existing works that deals with MTS forecasting. Nonetheless, little work has identified precisely the key bottleneck in this kind of problem. Herein, before formally proposing our solution, we start by systematically analyzing the problem to obtain greater insight. In real-world circumstances, we roughly classify an impact imposed on MTS into four classes in accordance with its activated ranges on the spatial and temporal dimensions. The four classes are composed of low-frequency local impact, low-frequency global impact, high-frequency local impact and high-frequency global impact. Here, "low-frequency" / "high-frequency" describes the activated range of the impact from the temporal view, and "global" / "local" describes the activated range from the spatial view. In particular, "low-frequency" means that the impact varies smoothly or, in other words, it tends to stay stable for a relatively long time; "high-frequency" means that the impact varies drastically; "global" means that the impact imposes a similar effect on all time series; "local" means that the impact only affects individual time series, or imposes different effects on different time series. Although the activated range on either the temporal or spatial dimension lies in a continuous spectrum, we consider only these four extreme cases as sufficient to reveal the essence of MTS. Any measurement of a time series is a mixture of four components respectively associated with the four classes of impacts, which can be formulated as follows:

$$\mathbf{X}_{i,t} = \mathbf{X}^{\text{lh}}_{i,t}\mathbf{X}^{\text{ll}}_{i,t}\mathbf{X}^{\text{gh}}_{t}\mathbf{X}^{\text{gl}}_{t} + const, \tag{1}$$

where $\mathbf{X}_{i,t} \in \mathbb{R}$ is the measurement of the i^{th} time series on time t, $\mathbf{X}^{\text{lh}}_{i,t} \in \mathbb{R}$ denotes the local high-frequency component, $\mathbf{X}^{\text{ll}}_{i,t} \in \mathbb{R}$ denotes the local low-frequency component, $\mathbf{X}^{\text{gh}}_{t} \in \mathbb{R}$ denotes the global high-frequency component and $\mathbf{X}^{\text{gl}}_{t} \in \mathbb{R}$ denotes the global low-frequency component. To understand this form of factorization more thoroughly, the following real-world example is used as the showcase. The time series data we present is the evolution of demand for a shared bike over three selected regions in New York City, as manifested in Fig. 1. In this example, the time of day serves as a global high-frequency impact; region identity, including regional population and functionality, serves as a local low-frequency impact; the day of week serves as a global low-frequency impact. Local high-frequency impacts are indistinguishable from the raw data, such as traffic accidents or congestion.

Time series forecasting is based mainly on its recent dynamics which is composed of contiguous measurements. Formally, the dynamics is expressed as a vector $\left[\mathbf{X}_{i,t}, \mathbf{X}_{i,t-1}, \cdots, \mathbf{X}_{i,t-\delta+1}\right]^{\top}$, where

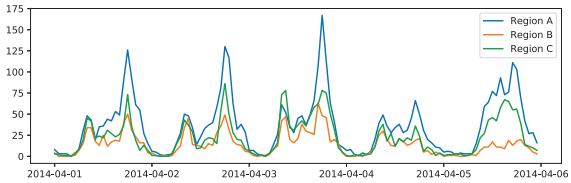


Figure 1: NYC shared bike demand.

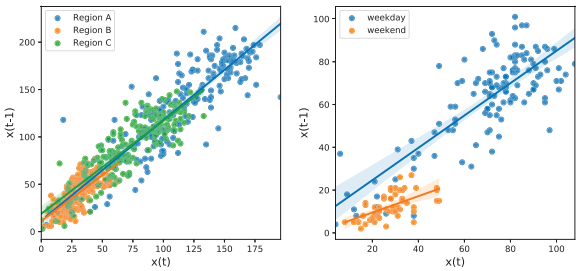
δ is the spanning time. However, canonical deep learning architectures employed in general time series forecasting tasks, such as LSTM [8], Transformer [12, 24] and Wavenet [23], only capture the directional information of this vector, which is a special type of temporal relationship, that results in discarding some informative components. To obtain the specific form of temporal relationship actually modeled, we presume two postulations which hold in the majority of real-world problems: (1) the low-frequency components (including both the global low-frequency and local low-frequency components) are stable over a given period; (2) the global high-frequency component well-dominate the local high-frequency component. Based on the factorization in Eq. (1) along with these two postulations, it is natural to derive the direction of the vector from the basis of its constant origin, the entry associated with historical time t_0 of which is calculated as follows:

$$\begin{aligned}\frac{\mathbf{X}_{i,t_0}}{\sqrt{\sum_{t'=0}^{\delta-1}(\mathbf{X}_{i,t-t'})^2}} &= \frac{\mathbf{X}^{\text{lh}}_{i,t_0}\mathbf{X}^{\text{ll}}_{i,t_0}\mathbf{X}^{\text{gh}}_{t_0}\mathbf{X}^{\text{gl}}_{t_0}}{\sqrt{\sum_{t'=0}^{\delta-1}(\mathbf{X}^{\text{lh}}_{i,t-t'})^2(\mathbf{X}^{\text{ll}}_{i,t-t'})^2(\mathbf{X}^{\text{gh}}_{t-t'})^2(\mathbf{X}^{\text{gl}}_{t-t'})^2}} \\ &\approx \frac{\mathbf{X}^{\text{gh}}_{i,t_0}}{\sqrt{\sum_{t'=0}^{\delta-1}(\mathbf{X}^{\text{gh}}_{i,t-t'})^2}},\end{aligned} \tag{2}$$

where the sign of each quantity is omitted for conciseness, as they do not influence our asserted conclusion. We note that the obtained directional vector merely accounts for the global high-frequency component, totally discarding the global low-frequency component, and its local-low frequency and local high-frequency counterparts.

Discarding the other three components incurs **spatial indistinguishability** and **temporal indistinguishability**. Spatial indistinguishability means that dynamics yielded by different variables are not adequately discernible. And temporal indistinguishability means that dynamics measured at specific times are not substantially discrete. For instance, looking at the three regions in Fig. 1, we consider their dynamics measured between 8pm and 9pm on different days, so they share the same global high-frequency element. In Fig. 2a, we plot the measurement at 8pm versus the measurement at 9pm over the three regions, where the data points are colored in accordance with their regional identities. Hence, a cluster of dynamics with the same color shares the identical local low-frequency component. In Fig. 2b, we only plot measurement pairs of region A, and separate them based on weekday or weekend. Here, a cluster of dynamics with the same color share the identical global low-frequency component. Different clusters of dynamics are supposed to be distinguishable, as their underlying local low-frequency components or global low-frequency components are disparate. However, the cluster-wise relationships (indicated by

the direction of a straight line fitting the intra-cluster data points) are highly correlated, which signifies either the spatial or the temporal indistinguishability. Such indistinguishability hinders deep neural networks from perceiving the spatial and temporal difference. It should be noted that, once the model tunes the fraction of parameters uniquely responsible for a cluster of dynamics, the forecast of other clusters will inversely degenerate in reaction to this action. This makes the current update prone to be counteracted by any subsequent updates, as the temporary status is not even locally optimal. Hence, the ultimate model mainly captures an average property of these clusters, emanating from the common global high-frequency component. This outcome also conforms completely to our deduction in Eq. (2).



(a) (b)

Figure 2: (a) Spatial indistinguishability; (b) Temporal indistinguishability.

1.2 Contributions

To address the above issues, the key is to refine more types of components from the original measurement. Thereby relationships that distinguish dynamics from the spatial view or the temporal view can be captured. In our work, we propose two kinds of normalization modules – temporal normalization (TN) and spatial normalization (SN) – which separately refine the high-frequency and local components. Specifically, the high-frequency component assists with distinguishing dynamics from the spatial view, and the local component facilitates differentiating the dynamics from the temporal view. With distinguishability on space and time, the model is able to exclusively fit each clusters of samples, especially some long-tailed samples. Moreover, we show the connection between our method and other state-of-the-art (SOTA) methods which rely on mutual relationship establishment to distinguish dynamics. We now have two prominent advantages, apart from the higher prediction accuracy: (1) the computational cost remains in $O(NT)$, rather than scaling up to $O(N^2T)$; (2) the converging speed is faster, as demonstrated by the experiments conducted.

2 RELATED WORK

2.1 Time Series Forecasting

Time series forecasting has been studied for decades. Traditional methods, such as ARIMA, can only learn linear relationship among different timesteps, which has an inherent deficiency in fitting many real-world time series data that are highly nonlinear. With the power of deep learning models, there is recently a large volume of work in this area that has achieved impressive performance. For instance, Qin et al. [15] adopt LSTM to capture the nonlinear dynamics and long-term dependencies in time series data. However, the memorizing capacity of LSTM is still restricted, as pointed out by Zhao et al. [32]. To resolve this issue, Tang et al. [21] create an external memory to explicitly store some representative patterns that can be frequently observed in the history, which is able to effectively guide the forecasting when similar patterns occur. Lai et al. [11] make use of skip connection to enable the information transmitting from distant history. Attention mechanism is another option to deal with the vanishing memory problem [6, 20]. Of these methods, Transformer is a representative architecture which consists of only attention operations [24]. To overcome the computation bottleneck of canonical Transformer, Li et al. [12] propose a novel mechanism that periodically skips some timesteps when performing attention. As far as we know, Wavenet [23] and TCN [2] are currently the most superior choices for modeling long-term time series data [19, 27, 28].

To tackle MTS, several studies [19, 30] assume that the multivariate time series data has a low-rank structure. Another thread of works [14, 15] leverage the attention mechanism to learn the correlations among individual time series. Recently, Wu et al. [27] inferred the inherent structure over the variables derived from self-learned encodings associated with each variable. The aforementioned methods make point estimation. Rangapuram et al. [16], Salinas et al. [17], Wang et al. [25] propose to deliver a confidence interval that is likely to contain the forthcoming observation.

2.2 Normalization

Normalization has been firstly adopted in deep image processing, and has fabulously promoted the performance of deep learning models in nearly all the tasks. There are multiple normalization methods, such as batch normalization [9], instance normalization [22] and group normalization [26], each of which is proposed to address a particular group of computer vision tasks. Of these, instance normalization has the greatest potential to be applied for our study, which was originally designed for image synthesis owing to its power to remove style information from the images. Researchers have found that feature statistics can capture the style of an image, and the remaining features upon normalizing the statistics are responsible for the content. Such a separable property enables the content of an image to be rendered in the style of another image, which also known as style transfer. The style information in the image is similar to the scale information in time series. Moreover, there is another line of work which explores the reason of why normalization trick facilitates the learning of deep neural networks [3, 4, 13, 18]. One major discovery of them is that normalization can increase the rankness of the feature space, or in other words, it enable the model to extract more diverse features.

3 PRELIMINARIES

In this section, we introduce the definitions and the assumption. All frequently used notations are reported in Table 1.

Table 1: Notations

Notation	Description
N, T_{in}, T_{out}	Number of variables / input steps / output steps.
$\mathbf{X} \in \mathbb{R}^{N \times T_{in}}$	Input data.
$\mathbf{Y} \in \mathbb{R}^{N \times T_{out}}$	Output data.
$\mathbf{Z} \in \mathbb{R}^{N \times T_{in} \times d_z}$	Latent data.
$\mathbf{Z}^{\text{lh}} \in \mathbb{R}^{N \times T_{in} \times d_z}$	Local high-frequency component.
$\mathbf{Z}^{\text{ll}} \in \mathbb{R}^{N \times T_{in} \times d_z}$	Local low-frequency component.
$\mathbf{Z}^{\text{gh}} \in \mathbb{R}^{T_{in} \times d_z}$	Global high-frequency component.
$\mathbf{Z}^{\text{gl}} \in \mathbb{R}^{T_{in} \times d_z}$	Global low-frequency component.
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	Vector or matrix that represents certain variable.
$+, \cdot, /$	Element-wise addition / multiplication / division.
$\overset{\text{i.i.d.}}{=}$	Two variables are i.i.d..
$*$	Placeholder.

Definition 1 (Time series forecasting)**.** Time series forecasting is formulated as the following conditional distribution:

$$P(\mathbf{Y}|\mathbf{X}) = \prod_{t=1}^{T_{out}} P(\mathbf{Y}_{:,t}|\mathbf{X}),$$

Definition 2 (Time series factorization)**.** Time series factorization generalizes Eq. (1) into the latent space, which takes the following form:

$$\mathbf{Z}_{i,t} = \mathbf{Z}^{\text{lh}}_{i,t}\mathbf{Z}^{\text{ll}}_{i,t}\mathbf{Z}^{\text{gh}}_{t}\mathbf{Z}^{\text{gl}}_{t}, \tag{3}$$

where:

$$\mathbf{Z}^{\text{lh}}_{i,t} \overset{\text{i.i.d.}}{=} \mathbf{Z}^{\text{lh}}_{i,t-1},\ \mathbf{Z}^{\text{ll}}_{i,t} \approx \mathbf{Z}^{\text{ll}}_{i,t-1},\ \mathbf{Z}^{\text{gh}}_{t} \overset{\text{i.i.d.}}{=} \mathbf{Z}^{\text{gh}}_{t-1},\ \mathbf{Z}^{\text{gl}}_{t} \approx \mathbf{Z}^{\text{gl}}_{t-1},\ \mathbf{Z}^{\text{l}^*}_{i,t} \overset{\text{i.i.d.}}{=} \mathbf{Z}^{\text{l}^*}_{j,t}.$$

Assumption 1. *The set of elements of* $\mathbf{Z}^{lh}_{i,t}, \mathbf{Z}^{ll}_{i,t}, \mathbf{Z}^{gh}_{t}$ *and* $\mathbf{Z}^{gl}_{t}$ *are mutually independent, which is formally written as:*

$$P(\mathbf{Z}^{ll}_{i,t}, \mathbf{Z}^{lh}_{i,t}, \mathbf{Z}^{gh}_{t}, \mathbf{Z}^{gl}_{t}) = \prod_{k=1}^{d_z} P(\mathbf{Z}^{ll}_{i,t,k}) P(\mathbf{Z}^{lh}_{i,t,k}) P(\mathbf{Z}^{gh}_{t,k}) P(\mathbf{Z}^{gl}_{t,k}). \tag{4}$$

4 METHODOLOGY

We display the overview of the architecture being leveraged in our work in Fig. 3. Some key variables with their shapes are labeled at their corresponding positions along the computation path. Generally, our framework follows a structure similar to Wavenet[2], except that we add both spatial normalization and temporal normalization modules which together are abbreviated as ST-Norm or STN.

4.1 Dilated Causal Convolution

In this section, we briefly introduce dilated causal convolution where the filter is applied with skipping values. For a 1-D signal $\mathbf{z} \in \mathbb{R}^T$ and a filter $f : \{0, \ldots, k-1\} \to \mathbb{R}$, the causal convolution on element t is defined as follows:

$$F(t) = (\mathbf{z} * f)(t) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{z}_{t-i}. \tag{5}$$

This formula can be easily generalized for multi-dimension signal but we omit its general form here for brevity. Moreover, padding (zero or replicate) with size of $k-1$ is appended to the left tail of the signal to ensure length consistency. We can stack multiple

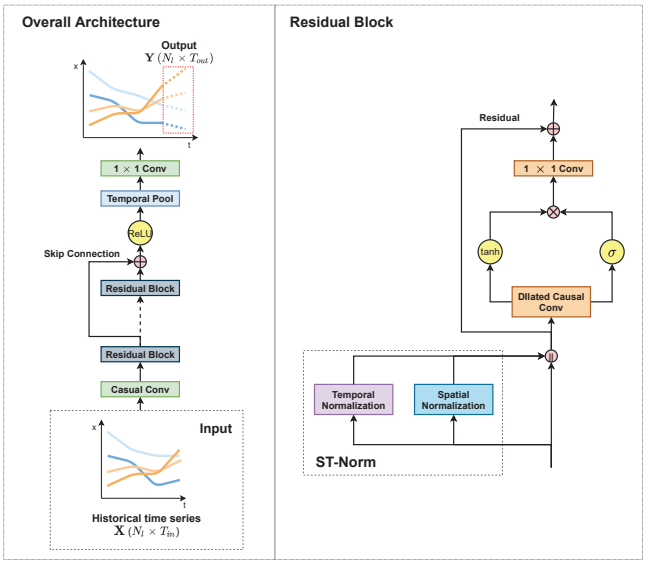


Figure 3: Overall architecture, where we just draw two residual blocks for illustration, but multiple blocks can be stacked layer by layer. +, × and || respectively denote element-wise addition, element-wise multiplication and concatenation

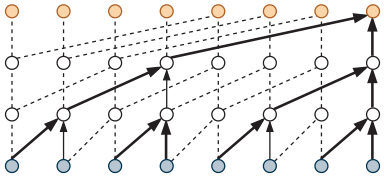


Figure 4: Dilated Causal Convolution.

causal convolution layers to obtain a larger receptive field for each element.

One shortcoming of using causal convolution is that either the kernel size or the number of layers increases in a linear manner with the range of the receptive field, and the linear relationship causes an explosion of parameters when modeling long history. Pooling is a natural choice to address this issue, but it sacrifices the order information presented in the signal. To this end, dilated causal convolution is leveraged, a form which supports the exponential expansion of the receptive field. The formal computing process is written as:

$$F(t) = (\mathbf{z} *_d f)(t) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{z}_{t-d \cdot i}, \tag{6}$$

where d is the dilation factor. Normally, d increases exponentially w.r.t. the depth of the network (i.e., 2^l at level l of the network). If d is 1 (2^0), then the dilated convolution operator $*_d$ reduces to a regular convolution operator $*$.

4.2 Temporal Normalization

Temporal normalization (TN) aims to refine the high-frequency components – both global and local – from the hybrid signal. Here, for conciseness, we introduce two notations to individually summarize high-frequency components and low-frequency components, which are expressed as:

$$\mathbf{Z}_{i,t}^{\text{high}} = \mathbf{Z}_{i,t}^{\text{lh}} \mathbf{Z}_{t}^{\text{gh}}, \ \mathbf{Z}_{i,t}^{\text{low}} = \mathbf{Z}_{i,t}^{\text{ll}} \mathbf{Z}_{t}^{\text{gl}}.$$

The applicability of TN is based on a reasonable assumption that the changing rates of low-frequency components are much slower than those of the high-frequency components. Or more technically, each low-frequency component approximately equals a constant over a period. Under this assumption, we are can apply TN on time series without the additional supplement of features characterizing the frequency. Such characteristic is well-suitable for an ample number of real-world problems, where the specific frequency is not available.

We start with expanding $\mathbf{Z}_{i,t}^{\text{high}}$ to obtain a desirable form whose distinct quantities can be derived from data:

$$\begin{aligned}
\mathbf{Z}_{i,t}^{\text{high}} &= \frac{\mathbf{Z}_{i,t}^{\text{high}} - E\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right]}{\sigma\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right] + \epsilon} \sigma\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right] + E\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right] \\
&= \frac{\mathbf{Z}_{i,t}^{\text{high}} \mathbf{Z}_{i,t}^{\text{low}} - \mathbf{Z}_{i,t}^{\text{low}} E\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right]}{\mathbf{Z}_{i,t}^{\text{low}} \sigma\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right] + \epsilon} \sigma\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right] + E\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right] \\
&= \frac{\mathbf{Z}_{i,t} - E\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right]}{(\pm)\sigma\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right] + \epsilon} \sigma\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right] + E\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right] \\
&= \frac{\mathbf{Z}_{i,t} - E\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right]}{\sigma\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right] + \epsilon} (\pm)\sigma\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right] + E\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right], \quad (7)
\end{aligned}$$

where ϵ is a small constant to preserve numerical stability; $\mathbf{Z}_{i,t}$ is observable; $E\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right]$ and $(\pm)\sigma\left[\mathbf{Z}_{i,t}^{\text{high}}|i\right]$ are mean and standard deviation (plus or minus) of the high-frequency impact on the i^{th} time series over time, which can be approximated by a pair of learnable vectors γ_i^{high} and β_i^{high} with the size of d_z. To estimate $E\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right]$ and $\sigma\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right]$ can be estimated as follows under Def. 2 and Assumption 1:

$$\begin{aligned}
E\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right] &\approx \frac{1}{\delta}\sum_{t'=1}^{\delta} \mathbf{Z}_{i,t-t'+1}^{\text{high}} \mathbf{Z}_{i,t}^{\text{low}} \\
&\approx \frac{1}{\delta}\sum_{t'=1}^{\delta} \mathbf{Z}_{i,t-t'+1}^{\text{high}} \mathbf{Z}_{i,t-t'+1}^{\text{low}} \\
&= \frac{1}{\delta}\sum_{t'=1}^{\delta} \mathbf{Z}_{i,t-t'+1} \quad (8) \\
\sigma^2\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right] &= E\left[\left(\mathbf{Z}_{i,t} - E\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right]\right)^2 \middle| \mathbf{Z}_{i,t}^{\text{low}}, i\right] \\
&\approx \frac{1}{\delta}\sum_{t'=1}^{\delta} \left(\mathbf{Z}_{i,t-t'+1}^{\text{high}} \mathbf{Z}_{i,t}^{\text{low}} - E\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right]\right)^2 \\
&\approx \frac{1}{\delta}\sum_{t'=1}^{\delta} \left(\mathbf{Z}_{i,t-t'+1}^{\text{high}} \mathbf{Z}_{i,t-t'+1}^{\text{low}} - E\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right]\right)^2 \\
&= \frac{1}{\delta}\sum_{t'=1}^{\delta} \left(\mathbf{Z}_{i,t-t'+1} - E\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right]\right)^2, \quad (9)
\end{aligned}$$

where δ is a period during which the low-frequency component approximately remains to be a constant. In our work, for simplicity, we let δ equal to the number of input time steps. By substituting the estimations of the four unobservable variables into Eq. (7), we are able to obtain the representation of the high-frequency component:

$$\mathbf{Z}_{i,t}^{\text{high}} = \frac{\mathbf{Z}_{i,t} - E\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right]}{\sigma\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{i,t}^{\text{low}}, i\right] + \epsilon} \gamma_i^{\text{high}} + \beta_i^{\text{high}} \quad (10)$$

Noticeably, TN has a close relationship with instance normalization (IN) for image data [22], where style plays the role of a low-frequency component and content serves as a high-frequency component. The novelty of our work is that we trace the origin of TN under the context of MTS, and deduce TN step-by-step from its origin.

4.3 Spatial Normalization

The objective of spatial normalization (SN) is to refine local components, composed of the local high-frequency component and the local low-frequency component. To achieve this objective, the primary task is first to eliminate global components, resulted from global impacts such as time of day, day of week and weather condition, etc. We also introduce two notations to summarize local and global components:

$$\mathbf{Z}_{t}^{\text{global}} = \mathbf{Z}_{t}^{\text{gh}} \mathbf{Z}_{t}^{\text{gl}}, \ \mathbf{Z}_{i,t}^{\text{local}} = \mathbf{Z}_{i,t}^{\text{lh}} \mathbf{Z}_{i,t}^{\text{ll}}.$$

Likewise, the applicability of SN is based on the assumption that the global impacts impose similar effects on all time series. For instance, in Fig. 1, there are common upward trends over the three regions with similar increasing rates when moving from 8am to 9am. Here, we need to clarify that we do not require the global impacts to strictly exert the same influence on each time series. Those effects that are not evenly observed on each time series could be complemented by the defined local component.

We firstly expand $\mathbf{Z}_{i,t}^{\text{local}}$ to an expression where each term can be approximated from data or be assigned with learnable parameters:

$$\begin{aligned}
\mathbf{Z}_{i,t}^{\text{local}} &= \frac{\mathbf{Z}_{i,t}^{\text{local}} - E\left[\mathbf{Z}_{i,t}^{\text{local}}|t\right]}{\sigma\left[\mathbf{Z}_{i,t}^{\text{local}}|t\right] + \epsilon} \sigma\left[\mathbf{Z}_{i,t}^{\text{local}}|t\right] + E\left[\mathbf{Z}_{i,t}^{\text{local}}|t\right] \\
&= \frac{\mathbf{Z}_{i,t}^{\text{local}} \mathbf{Z}_{t}^{\text{global}} - \mathbf{Z}_{t}^{\text{global}} E\left[\mathbf{Z}_{i,t}^{\text{local}}|t\right]}{\mathbf{Z}_{t}^{\text{global}} \sigma\left[\mathbf{Z}_{i,t}^{\text{local}}|t\right] + \epsilon} \sigma\left[\mathbf{Z}_{i,t}^{\text{local}}|t\right] + E\left[\mathbf{Z}_{i,t}^{\text{local}}|t\right] \\
&= \frac{\mathbf{Z}_{i,t} - E\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{t}^{\text{global}}, t\right]}{(\pm)\sigma\left[\mathbf{Z}_{i,t}|\mathbf{Z}_{t}^{\text{global}}, t\right] + \epsilon} \sigma\left[\mathbf{Z}_{i,t}^{\text{local}}|t\right] + E\left[\mathbf{Z}_{i,t}^{\text{local}}|t\right]
\end{aligned}$$

$$= \frac{\mathbf{Z}_{i,t} - E\left(\mathbf{Z}_{i,t} \middle| \mathbf{Z}_t^{\text{global}}, t\right)}{\sigma\left(\mathbf{Z}_{i,t} \middle| \mathbf{Z}_t^{\text{global}}, t\right) + \epsilon} (\pm)\sigma\left(\mathbf{Z}_{i,t}^{\text{local}} \middle| t\right) + E\left(\mathbf{Z}_{i,t}^{\text{local}} \middle| t\right) \tag{11}$$

where $\mathbf{Z}_{i,t}$ is directly observable; $(\pm)\sigma\left(\mathbf{Z}_{i,t}^{\text{local}} \middle| t\right)$ and $E\left(\mathbf{Z}_{i,t}^{\text{local}} \middle| t\right)$ are approximated by two learnable vectors[2] γ^{local} and β^{local}; the estimation of $E\left(\mathbf{Z}_{i,t} \middle| \mathbf{Z}_t^{\text{global}}, t\right)$ and $\sigma\left(\mathbf{Z}_{i,t} \middle| \mathbf{Z}_t^{\text{global}}, t\right)$ can be derived from data in the following ways under Def. 2 and Assumption 1:

$$\begin{aligned} E\left(\mathbf{Z}_{i,t} \middle| \mathbf{Z}_t^{\text{global}}, t\right) &\approx \frac{1}{N}\sum_{j=1}^{N} \mathbf{Z}_{j,t}^{\text{local}} \mathbf{Z}_t^{\text{global}} \\ &= \frac{1}{N}\sum_{j=1}^{N} \mathbf{Z}_{j,t} \end{aligned} \tag{12}$$

$$\begin{aligned} \sigma^2\left(\mathbf{Z}_{i,t} \middle| \mathbf{Z}_t^{\text{global}}, t\right) &= E\left[\left(\mathbf{Z}_{i,t} - E\left(\mathbf{Z}_{i,t} \middle| \mathbf{Z}_t^{\text{global}}, t\right)\right)^2 \middle| \mathbf{Z}_t^{\text{global}}, t\right] \\ &\approx \frac{1}{N}\sum_{j=1}^{N} \left(\mathbf{Z}_{j,t}^{\text{local}} \mathbf{Z}_t^{\text{global}} - E\left(\mathbf{Z}_{i,t} \middle| \mathbf{Z}_t^{\text{global}}, t\right)\right)^2 \\ &= \frac{1}{N}\sum_{j=1}^{N} \left(\mathbf{Z}_{j,t} - E\left(\mathbf{Z}_{i,t} \middle| \mathbf{Z}_t^{\text{global}}, t\right)\right)^2 \end{aligned} \tag{13}$$

By substituting the approximations of the four unobservable variables into Eq. (11), we are able to obtain the composite representation of the local components:

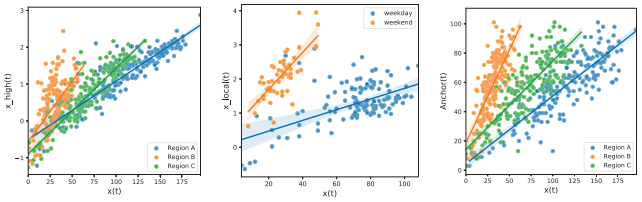
$$\mathbf{Z}_{i,t}^{\text{local}} = \frac{\mathbf{Z}_{i,t} - E\left(\mathbf{Z}_{i,t} \middle| \mathbf{Z}_t^{\text{global}}, t\right)}{\sigma\left(\mathbf{Z}_{i,t} \middle| \mathbf{Z}_t^{\text{global}}, t\right) + \epsilon} \gamma^{\text{local}} + \beta^{\text{local}} \tag{14}$$

SN is a counterpart of TN in the spatial domain, where high-frequency components act as local components, and low-frequency components correspond to global components. By distilling the local or high-frequency components from the original signal, the model can capture fine-grained variation, which is extraordinarily instrumental in time series forecasting.

4.4 Forecasting and Learning

We let $\mathbf{Z}^{(L)} \in \mathbb{R}^{N_l \times T_{in} \times d_z}$ denote the output from the last residual block, where each row $\mathbf{z}^{(L)} \in \mathbb{R}^{T_{in} \times d_z}$ represents a variable. Then, we employ a temporal pooling block to perform temporal aggregation for each variable. Several types of pooling operations can be applied, such as max pooling and mean pooling, depending on the problem being studied. In our case, we select the vector in the most recent time slot as the pooling result, which is treated as the representation of the entire signal. Finally, we make a separate prediction for each variable, based on the obtained representation by a shared fully connected layer.

In the learning phase, our objective is to minimize the mean squared error between the predicted values and ground truth values. In addition, we use the Adam optimizer [10] to optimize this target.



(a) TN (b) SN (c) Graph-based

Figure 5: Relationships produced by the three operations.

4.5 Discussion

To illustrate how TN and SN reframe the feature space, we apply them over the raw input data, and examine whether they mitigate the issues we raise in Fig. 2. We plot the original quantity versus the temporally normalized quantity in Fig. 5a, and the original quantity versus the spatially normalized quantity in Fig. 5b. It is apparent that the pairwise relationship between the original quantity and the temporally normalized quantity separates different regions, and the pairwise relationship between the original quantity and the spatially normalized quantity separates different days.

A few SOTA approaches [1, 27, 28] propose to establish a mutual relationship between different time series in order to refine the local component. In essence, they contrast a pair of time series which share the same global components over time, thereby allowing the local component of individual time series to be highlighted. For instance, we contrast the three time series in Fig. 1 with a single time series (regarded as an anchor), which results in a pairwise relationship reflecting the identity of each time series as shown in Fig. 5c. However, the eligible anchors are often unknown, and different time series may need to be paired with different anchors. To automatically identify the anchor for each time series, these methods employ a graph-learning module to explore every possible pair of time series. Their computational complexity is $O(TN^2)$. Unlike other approaches proposed in this area, the normalization modules involved in our method only require $O(TN)$ operations.

5 EVALUATION

In this section, we conduct extensive experiments on three common datasets to validate the effectiveness of ST-Norm from different aspects.

5.1 Experimental Setting

5.1.1 Datasets. We validate our model on three real-world datasets, including BikeNYC, PeMSD7 and Electricity. The statistics regarding each dataset as well as the corresponding settings of the designed task are reported in Table 6. More details can be found in Appendix A.1. We standardize the values in each dataset to facilitate training and transform them back to the original scale in the testing phase.

5.1.2 Network Setting. We add an instance normalization (IN) module [22] in parallel with SN and TN as another complement[3]. The

[2] Each time would possess the identical prior distribution if dynamic laws, such as periodicity, is unknown.

[3] The implementation can be found in our code.

batch size is 4, and the input length of the batch sample is 16. For the Wavenet backbone, the layer number is set to 4, the kernel size of each DCC component is 2, and the associated dilation rate is 2^i, where i is the index of the layer (counting from 0). Such settings collectively enable the output from Wavenet to perceive 16 input steps. The number of hidden channels d_z in each DCC is 16. We apply zero-padding on the left tail of the input to enable the length of the output from DCC to equal to 16 as well. The learning rate of the Adam optimizer is 0.0001.

5.1.3 Evaluation Metrics. We validate our model by root mean squared error (RMSE), mean absolute error (MAE) and mean absolute percentage error (MAPE). We repeat the experiment ten times for each model on each dataset and report the mean of the results.

5.2 Baseline Models

- **MTGNN** [27]. MTGNN constructs inter-variate relationships by introducing a graph-learning module. Specifically, the graph learning module connects each hub node with its top k nearest neighbors in a defined metric space. MTGNN's backbone architecture for temporal modeling is Wavenet.
- **Graph Wavenet** [28]. The architecture of Graph Wavenet is like MTGNN. The major difference is that the former derives a soft graph where each pair of nodes has a continuous probability of being connected.
- **AGCRN** [1]. AGCRN also equips with a graph-learning module to establish inter-variate relationship. Furthermore, it uses a personalized RNN to model the temporal relationship for each time series.
- **Transformer** [12]. This model captures the long-term dependencies in time series data through using an attention mechanism, where the keys and queries are yielded by causal convolution over local context to model segment-level correlation.
- **LSTNet** [11]. There are two components in LSTNet: one is a conventional autoregressive model, and the other is an LSTM with an additional skip connection over the temporal dimension.
- **TCN**. [2] The architecture of TCN is like Wavenet, except that the nonlinear transformation in each residual block is made up of two rectified linear units (ReLU).

We also test the performance of TCN and Transformer incorporating STN, where STN is similarly applied before the causal convolution operation in each layer.

5.3 Experiment Results

Table 2: Performance on the BikeNYC dataset

Models	1 hour			2 hour			3 hour		
	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE
LSTNet	19.6%	2.55	5.35	21.1%	2.77	6.04	22.6%	2.99	6.63
AGCRN	17.3%	2.34	4.76	18.7%	2.56	5.51	20.3%	2.77	6.07
Graph Wavenet	18.0%	2.39	4.78	19.4%	2.65	5.53	20.8%	2.86	6.05
MTGNN	19.0%	2.55	5.05	21.1%	2.88	6.00	22.9%	3.13	6.61
Transformer	22.8%	2.98	6.16	27.1%	3.66	7.88	29.7%	4.12	8.95
Transformer + STN	17.7%	2.36	4.75	19.1%	2.57	5.51	20.7%	2.78	6.09
TCN	22.4%	2.90	5.98	26.4%	3.57	7.66	29.1%	4.07	8.76
TCN + STN	**16.8%**	2.30	4.51	18.6%	2.55	5.34	20.4%	2.77	5.92
Wavenet	22.1%	2.86	5.92	26.3%	3.52	7.58	29.0%	3.97	8.68
Wavenet + STN	16.9%	**2.23**	**4.48**	**18.3%**	**2.47**	**5.28**	**20.1%**	**2.68**	**5.88**
Improvements	+2.8%	+4.7%	+5.8%	+2.1%	+3.5%	+4.1%	+0.9%	+3.2%	+2.8%

Table 3: Performance on the PeMSD7 dataset.

Models	30 min			60 min			90 min		
	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE
LSTNet	8.10%	3.88	6.52	8.43%	4.01	6.71	9.06%	4.30	7.12
AGCRN	4.87%	2.34	4.24	6.48%	3.07	5.58	7.27%	3.43	6.19
Graph Wavenet	4.90%	2.33	4.25	6.77%	3.19	5.69	7.57%	3.57	6.25
MTGNN	5.18%	2.46	4.48	7.61%	3.57	6.31	9.03%	4.25	7.26
Transformer	5.82%	2.75	5.03	9.31%	4.34	7.51	11.8%	5.49	9.02
Transformer + STN	4.86%	2.33	4.26	6.50%	3.08	5.65	7.33%	3.48	6.31
TCN	5.80%	2.75	4.97	9.44%	4.43	7.53	12.0%	5.61	9.06
TCN + STN	4.91%	2.34	4.22	6.42%	3.04	5.51	7.12%	3.38	6.05
Wavenet	5.50%	2.61	4.80	8.75%	4.10	7.20	11.0%	5.16	8.61
Wavenet + STN	**4.71%**	**2.25**	**4.12**	**6.23%**	**2.95**	**5.48**	**6.97%**	**3.29**	**6.01**
Improvements	+3.2%	+3.4%	+3.0%	+3.8%	+3.9%	+1.7%	+4.1%	+4.0%	+2.9%

Table 4: Performance on the Electricity dataset.

Models	1 hour			2 hour			3 hour		
	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE
LSTNet	22.4%	31.1	61.2	23.0%	31.8	62.6	24.8%	33.8	66.8
AGCRN	12.2%	17.1	36.3	16.1%	22.3	49.1	19.1%	26.0	56.6
Graph Wavenet	10.9%	15.9	34.9	15.8%	22.4	49.9	19.1%	26.5	57.7
MTGNN	11.1%	15.8	32.5	16.0%	22.2	46.3	19.6%	26.5	54.6
Transformer	11.2%	16.6	36.3	17.6%	25.4	53.7	22.2%	31.8	65.0
Transformer + STN	13.2%	17.4	35.9	17.7%	23.5	48.8	21.2%	27.9	58.0
TCN	11.1%	16.3	35.5	17.3%	25.0	52.4	21.5%	30.7	62.0
TCN + STN	13.2%	16.7	31.7	16.5%	21.5	42.8	20.1%	25.4	50.9
Wavenet	**10.8%**	15.8	33.3	16.8%	23.8	49.5	21.1%	29.5	60.3
Wavenet + STN	12.0%	**15.6**	**30.9**	**15.1%**	**20.1**	**42.2**	**17.1%**	**23.0**	**49.2**
Improvements	-11.0%	+1.2%	+4.9%	+4.4%	+9.4%	+8.8%	+10.4%	+11.5%	+9.8%

The experimental results on the BikeNYC, PeMSD7 and Electricity datasets are separately reported in Table 2, Table 3 and Table 4. The improvements achieved by Wavenet + STN over the best benchmarks are recorded in the last row of each table.

It is obvious that Wavenet + STN achieves SOTA results over almost all horizons on BikeNYC, PeMSD7 and Electricity data. The reason is that we refine the high-frequency components from both the temporal view and the spatial view, which are generally overlooked by baseline models. Next, we reveal the cause of Wavenet + STN's under-performance on the electricity dataset over the first horizon with respect to MAPE. As shown in Fig. 9b, electricity data follows a long-tailed distribution – there is a certain portion of quantities exceeding a relatively high level. Recall that the optimization targets minimizing mean squared error, which means that more weights are placed on large errors. Moreover, every sample is treated equivalently in the estimation of global statistics. Therefore, the model can fit long-tailed samples better, but at the cost of degrading the fitness on normal samples.

We also display the process of loss convergence in Appendix A.2. It shows that with the additional STN module, the converging speeds of the models are accelerated by a large margin, faster than that of nearly all baseline models.

5.4 Ablation Study

Table 5: Ablation Study

		GSTN	STN	Graph	TN	SN	Vanilla
B	RMSE	**5.18**	5.21	5,46	5.63	6.37	7.40
	MAE	**2.45**	2.47	2.63	2.64	2.99	3.44
	MAPE	**18.4%**	18.7%	19.4%	19.8%	22.6%	25.8%
P	RMSE	**5.16**	5.22	5.40	5.35	6.12	6.87
	MAE	**2.77**	2.83	3.03	2.90	3.47	3.96
	MAPE	**5.86%**	5.97%	6.41%	6.08%	7.41%	8.43%
E	RMSE	**38.9**	40.8	47.5	44.1	45.9	47.9
	MAE	**18.9**	19.6	21.6	21.4	22.6	23.1
	MAPE	**14.4%**	14.7%	15.3%	16.2%	16.7%	15.9%

To validate the effectiveness of SN and TN, we design several variants as follows. We also investigate whether a graph-learning

module complements STN by testing a variant containing them both. As all the variants contain the standard Wavenet backbone, we omit Wavenet in the name for brevity.

- **GSTN**. STN with an adaptive graph learning module as in Graph Wavenet.
- **Graph**. Graph Wavenet.
- **SN**. Wavenet with SN module.
- **TN**. Wavenet with TN module.

We evaluate these variants on all the three datasets and report the overall results in Table 5. It is evident that both of SN and TN contribute to the enhancement. Moreover, with an adaptive graph-learning module, the performance of STN rises marginally. We can conclude that STN largely substitutes and surpasses the graph-learning module.

5.5 Hyper-parameter Analysis

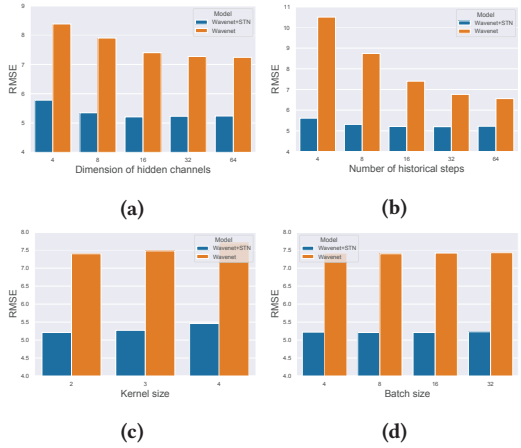


Figure 6: Hyper-parameter analysis.

We further study the effect of different settings of the hyper-parameters in the proposed modules. There are four hyper-parameters to be manually set by practitioners, consisting of the dimension of hidden channels d_z, the number of historical steps input to the model, the kernel size of DCC and the batch size. The study results are reported in Fig. 6, from which we are able to draw a major conclusion: STN not only boosts the performance, but also increases the stability of the performance under different hyper-parameter settings.

5.6 Case Study

The time series data we leveraged for the case study is BikeNYC. For each of SN and TN, we examine three representative regions at specified times to reflect what the module extracts from data. We collect the intermediate representations output from the two normalization modules installed in the top residual block and compress them via t-SNE for the sake of visualization. For comparison, we also examine their associated input representations, each of which is a concatenation of raw measurements. Next, we will discuss separately the outcomes of the two modules in details.

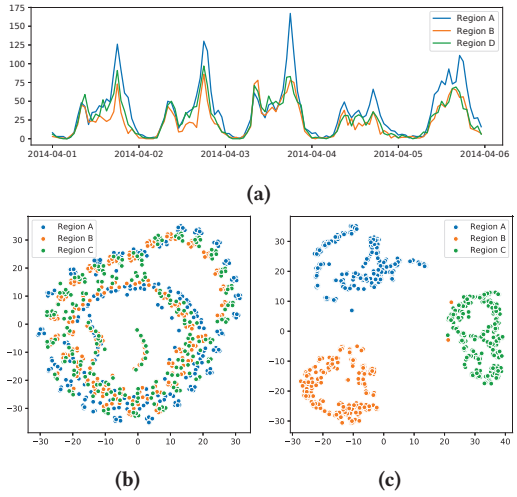


Figure 7: Case study on SN.

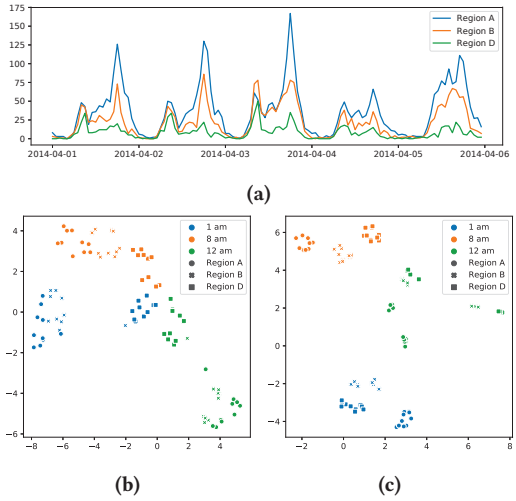


Figure 8: Case study on TN.

5.6.1 Spatial Normalization. SN removes the global component $\mathbf{Z}_t^{\text{global}}$ from the original measurement, while retaining the local component $\mathbf{Z}_{i,t}^{\text{local}}$. In Fig. 7a, we display the demand evolution during a given period over the three investigated regions. We can observe that the three regions have similar evolution patterns, especially regions B and C. The representations concatenated by original measurements are plotted in Fig. 7b, and the intermediate representations output from SN are plotted in Fig. 7c. We can observe that SN completely rearranges the representations in accordance with its regional identity. This observation demonstrates that the low-frequency parts in the local components are roughly invariant within the group belonging to the same region. This coincides with our understanding that some regional attributes, such as the population and the functionality, are stable over time.

5.6.2 Temporal Normalization. TN attempts to eliminate the low-frequency component $\mathbf{Z}_{i,t}^{\text{low}}$, while highlighting the high-frequency component $\mathbf{Z}_{i,t}^{\text{high}}$. To reflect the characteristics of the representations output from TN, we take another D into consideration, as shown in Fig. 8a. Noticeably, the magnitude of the demand over region D is substantially smaller than those over regions A or B. Here, we account for three different times in a day, consisting of 1am, 8am and 12pm. Likewise, the input representations are plotted in Fig. 8b, and the intermediate representations in Fig. 8c. As shown in Fig. 8b, instances belonging to region D are mixed up without separation between different times, which signifies that the model will struggle to differentiate the times those instances occurred. By contrast, TN mitigates this issue as it forms clusters of the instances with the same occurrence time.

6 CONCLUSION

In this work, we introduce a novel way to factorize MTS data. Following factorization, we propose temporal normalization and spatial normalization, which respectively refine the high-frequency component and the local component from the MTS data. The experimental results show the effectiveness and efficiency of these two modules.

7 ACKNOWLEDGMENTS

This work was supported in part by ARC under Grant DP180100106 and DP200101328.

REFERENCES

[1] LEI BAI, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. *Advances in Neural Information Processing Systems* 33 (2020).

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[3] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. 2018. Understanding batch normalization. In *Advances in Neural Information Processing Systems*. 7694–7705.

[4] Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. 2020. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. In *Advances in Neural Information Processing Systems*, Vol. 33. 18387–18398.

[5] Daizong Ding, Mi Zhang, Xudong Pan, Min Yang, and Xiangnan He. 2019. Modeling extreme events in time series prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1114–1122.

[6] Chenyou Fan, Yuze Zhang, Yi Pan, Xiaoyue Li, Chi Zhang, Rong Yuan, Di Wu, Wensheng Wang, Jian Pei, and Heng Huang. 2019. Multi-horizon time series forecasting with temporal attention learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2527–2535.

[7] Xu Geng, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu. 2019. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3656–3663.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[9] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.

[10] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[11] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 95–104.

[12] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*. 5243–5253.

[13] Xiangru Lian and Ji Liu. 2019. Revisit Batch Normalization: New Understanding and Refinement via Composition Optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*. 3254–3263.

[14] Yuxuan Liang, Songyu Ke, Junbo Zhang, Xiuwen Yi, and Yu Zheng. 2018. Geoman: Multi-level attention networks for geo-sensory time series prediction.. In *IJCAI*. 3428–3434.

[15] Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison W Cottrell. 2017. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2627–2633.

[16] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. 2018. Deep state space models for time series forecasting. In *Advances in neural information processing systems*. 7785–7794.

[17] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2019. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* (2019).

[18] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. 2018. How does batch normalization help optimization?. In *Advances in Neural Information Processing Systems*. 2483–2493.

[19] 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*. 4837–4846.

[20] Qingxiong Tan, Mang Ye, Baoyao Yang, Siqi Liu, Andy Jinhua Ma, Terry Cheuk-Fung Yip, Grace Lai-Hung Wong, and PongChi Yuen. 2020. DATA-GRU: Dual-Attention Time-Aware Gated Recurrent Unit for Irregular Multivariate Time Series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 930–937.

[21] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Charu C Aggarwal, Prasenjit Mitra, and Suhang Wang. 2020. Joint Modeling of Local and Global Temporal Dynamics for Multivariate Time Series Forecasting with Missing Values.. In *AAAI*. 5956–5963.

[22] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. 2016. Instance Normalization: The Missing Ingredient for Fast Stylization. *CoRR* abs/1607.08022 (2016). arXiv:1607.08022 http://arxiv.org/abs/1607.08022

[23] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. [n.d.]. WaveNet: A Generative Model for Raw Audio. In *9th ISCA Speech Synthesis Workshop*. 125–125.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[25] Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. 2019. Deep factors for forecasting. In *International Conference on Machine Learning*. PMLR, 6607–6617.

[26] Yuxin Wu and Kaiming He. 2018. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*. 3–19.

[27] 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 the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 753–763.

[28] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 1907–1913.

[29] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*.

[30] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. 2016. Temporal regularized matrix factorization for high-dimensional time series prediction. In *Advances in neural information processing systems*. 847–855.

[31] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 1655–1661.

[32] Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. 2020. Do rnn and lstm have long memory?. In *International Conference on Machine Learning*. PMLR, 11365–11375.

A APPENDIX

A.1 Data

Table 6: Dataset statistics.

Tasks	Electricity	PeMSD7	BikeNYC
Start time	10/1/2014	5/1/2012	4/1/2014
End time	12/31/2014	6/30/2012	9/30/2014
Sample rate	1 hour	30 minutes	1 hour
# Timesteps	2184	2112	4392
# Variate	336	228	128
Training size	1848	1632	3912
Validation size	168	240	240
Testing size	168	240	240
Output length	3	3	3

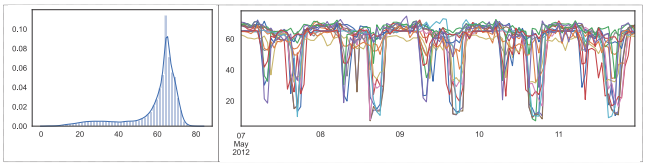
In Table 6, statistics of the datasets are reported. More details regarding the datasets are introduced below.

- PeMSD7 [29]. The data is collected from Caltrans Performance Measurement System (PeMS) by sensor stations, which are deployed to monitor traffic speed across the major metropolitan areas of the California state highway system. We further aggregate the data to 30-minute interval by average pooling.
- Electricity[4]. The original dataset contains the electricity consumption of 370 points/clients, from which 34 outlier points that contain extreme values are removed. Moreover, we calculate the hourly average consumption for each point, and take it as the time series being modeled.
- BikeNYC [31]. Each time series in this dataset denotes the aggregate demand for shared bikes over a region in New York City. We do not consider the spatial relationship presented in the PeMSD7 and BikeNYC data, since our objective is to study the temporal patterns.

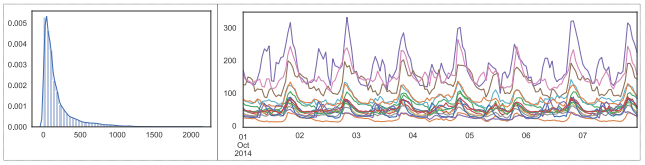
Furthermore, we display the data distribution and several exemplar time series in Fig. 9 to gain more insights from each dataset. We can observe that each of the three types of data lays in a wide range of scale, and exhibits periodicity to some extent. However, their evolving patterns are entirely different where the electricity time series shows the greatest diversity.

A.2 Loss Convergence

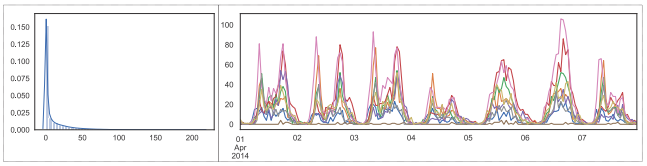
[4]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014



(a) PeMSD7.

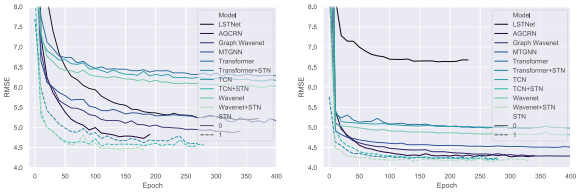


(b) Electricity.

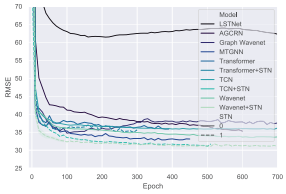


(c) BikeNYC.

Figure 9: For each (a), (b) and (c), the left figure shows the probability density function of observed values aggregated from all variables at all time steps, and the right figure displays some sample time series.



(a) BikeNYC (b) PeMSD7



(c) Electricity

Figure 10: Loss convergence.