Towards Spatio-Temporal Aware Traffic Time Series Forecasting–Full Version

Razvan-Gabriel Cirstea[1], Bin Yang[1]*, Chenjuan Guo[1], Tung Kieu[1], Shirui Pan[2]

[1]Department of Computer Science, Aalborg University, Denmark
[2]Faculty of Information Technology, Monash University, Australia
{razvan, tungkvt, cguo, byang}@cs.aau.dk, shirui.pan@monash.edu

Abstract—Traffic time series forecasting is challenging due to complex *spatio-temporal dynamics*—time series from different locations often have distinct patterns; and for the same time series, patterns may vary across time, where, for example, there exist certain periods across a day showing stronger temporal correlations. Although recent forecasting models, in particular deep learning based models, show promising results, they suffer from being *spatio-temporal agnostic*. Such spatio-temporal agnostic models employ a shared parameter space irrespective of the time series locations and the time periods and they assume that the temporal patterns are similar across locations and do not evolve across time, which may not always hold, thus leading to sub-optimal results. In this work, we propose a framework that aims at turning spatio-temporal agnostic models to *spatio-temporal aware* models. To do so, we encode time series from different locations into stochastic variables, from which we generate location-specific and time-varying model parameters to better capture the spatio-temporal dynamics. We show how to integrate the framework with canonical attentions to enable spatio-temporal aware attentions. Next, to compensate for the additional overhead introduced by the spatio-temporal aware model parameter generation process, we propose a novel window attention scheme, which helps reduce the complexity from quadratic to linear, making spatio-temporal aware attentions also have competitive efficiency. We show strong empirical evidence on four traffic time series datasets, where the proposed spatio-temporal aware attentions outperform state-of-the-art methods in term of accuracy and efficiency. This manuscript provides a full version of [1].

I. INTRODUCTION

We are witnessing impressive technological developments in the last years, leading to inexpensive and effective monitoring systems using a wide variety of sensors [2], [3]. For example, the transportation sector has deployed different sensors, e.g., loop detectors or speed cameras, in different roads to continuously capture useful traffic information, such as speeds and traffic flows, at different roads, giving rise to a large amount of traffic time series. An example is shown in Figure 1, where we plot the physical locations of four traffic flow sensors deployed on the highways of Sacramento, California, and the traffic flow time series collected by these sensors.

Traffic forecasting using traffic time series collected by the sensors is a core component of many Intelligent Transportation Systems (ITS) [4]. Accurate forecasting on traffic statuses, e.g., traffic flow, speeds, lane occupancy, reveals holistic dynamics of the underlying traffic network, contributing to
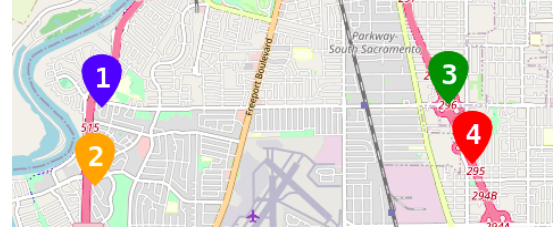
(a) Sensors Locations, Sacramento, California.
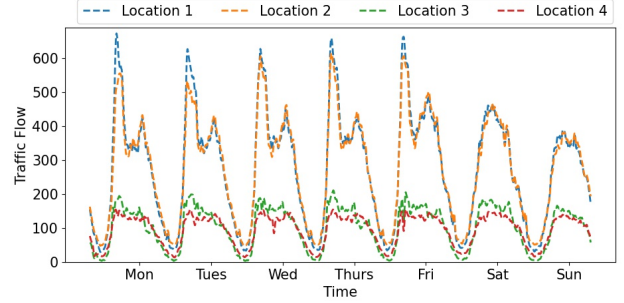
(b) Time series from the above 4 sensors, showing 1-week traffic conditions

Fig. 1: Example of four sensors and their time series. The time series from sensors 1 and 2 show similar traffic patterns as the sensors are deployed on the same street. The same happens for sensors 3 and 4, but the time series from sensors 1 and 3 show different patterns.

early warnings for emergency management [5], [6], diagnosing faulty equipment [7]–[9], and route planning and recommendation [10]–[14]. However, accurate traffic forecasting is challenging due to its complex *spatio-temporal dynamics*.

First, time series collected from different locations have distinct patterns. To illustrate this, we show four different time series collected from the four sensors shown in Figure 1(a). Sensors 1 (blue) and 2 (orange), which are deployed along one street, exhibit different patterns, comparing to sensors 3 (green) and 4 (red), which are deployed along another street. More specifically, sensors 1 and 2 have two clear peek hours in the morning and in the afternoon, respectively, in weekdays; while sensors 3 and 4 do not have a spike in the afternoon, instead the flow gradually decreases. This calls for *spatial-aware modeling* where different model parameters can be used for modeling time series from different locations.

Second, the patterns within time series often vary across

* Corresponding Author

1

time. Figure 1(a) shows that during weekdays, it exhibits similar temporal patterns, which are different from the patterns during the weekends. This holds for both sets of sensors. In such scenarios, it can be beneficial to automatically adjust model parameters across time such that they can better model different temporal patterns in, e.g., weekdays vs. weekends. The ability of adjusting model parameters across time can also be useful for accidents or road closures, where traffic patterns may deviate from regular temporal patterns. This calls for *temporal-aware modeling* where different model parameters can be used for modeling time series during different times.

Based on the above, the complex spatio-temporal traffic dynamics calls for spatio-temporal aware forecasting model that is able to (1) capture distinct patterns at different locations; (2) quickly adapt to pattern changes in different time periods.

However, existing methods are often *spatio-temporal agnostic*, thus failing to capture the spatio-temporal dynamics. Traditional methods such as Auto-Regressive Integrated Moving Average (ARIMA) or Vector Auto-Regression (VAR) cannot capture nonlinear patterns and thus fail to capture complex spatio-temporal patterns among different time series, resulting in sub-optimal forecasting accuracy [15]. Recently, various deep learning models have been proposed with promising accuracy, which are mainly based on three types of architectures, i.e., recurrent neural networks (RNNs) [16]–[20], temporal convolution networks (TCNs) [21]–[23], and attentions [24]–[26]. However, none of the architectures offers time-varying and location-specific modeling. First, the model parameters are static, thus failing to capture patterns that change over time and being *temporal-agnostic*. In particular, the weight matrices in RNN cells, the convolution filters in TCNs, and the projection matrices in attentions, stay the same across time [16], [17], [22], [27]–[29]. Second, the same parameters are employed for time series from different locations [16], [17], [22], [29], [30], by assuming that the traffic patterns are similar for all locations. However, this is often not the case in reality as shown in Figure 1. Such modeling forces the learned parameters to represent an "average" of the traffic patterns among all time series, thus being *spatial-agnostic*.

In this paper, we strive to enable *spatio-temporal aware* forecasting models by utilizing time-varying and location-specific model parameters in a data driven manner. We propose to encode each location's recent time series into a stochastic variable that captures both location-specific and time-varying patterns, from which we generate distinct model parameters for modeling time series from different locations and for different time periods. This approach is purely data driven, which avoids using additional location and time specific features, such as coordinates of sensor locations and prior knowledge on peak vs. offpeak hours. This approach is generic as it can be applied to generate spatio-temporal aware parameters for different models, such as RNNs, TCNs and attentions. We use attention as a concrete model in this paper as it shows superior accuracy, even for very long time series [26], compared to, e.g., RNNs and TCNs.

To alleviate the additional overhead caused by the spatio-temporal aware model parameter generation, we propose a novel efficient and accurate attention mechanism to ensure that the spatio-temporal aware attentions also have competitive efficiency. More specifically, in canonical self-attention, each timestamp directly attends all other timestamps, thus resulting in quadratic complexity. We propose a *window attention* to reduce the complexity from quadratic to linear, thus ensuring competitive efficiency overall, without compromising accuracy. More specifically, we strategically break an input time series into small windows and introduce a small constant number of *proxies* for each window to harvest information within the window. Then, each timestamp only attends to the proxies, but not all other timestamps anymore. This enables us to reduce the complexity from $\mathcal{O}(H^2)$ of the canonical self-attention to $\mathcal{O}(H)$ of the window attention, where H is the length of the input time series. This linear window attention operation ensures competitive overall efficiency, even with the additional spatio-temporal aware parameter generation.

To the best of our knowledge, this is the first study to enable spatio-temporal aware modeling where time-varying and location-specific model parameters are employed. The study makes four contributions. First, we propose Spatio-Temporal Aware Parameter Generation Network to generate *location-specific* and *time-varying* model parameters to turn spatio-temporal agnostic models to spatio-temporal aware models. Second, we integrate the proposed mechanism into attentions to enable spatio-temporal aware attentions. Third, we propose an efficient window attention by using proxies, reducing the complexity of attentions operations from quadratic to linear, thus ensuring overall competitive efficiency for spatio-temporal aware attentions. Fourth, we conduct extensive experiments on four commonly used traffic time series data sets, justifying the design choices and demonstrating that the proposal outperforms the state-of-the-art.

Paper Outline: Section II covers the recent advancements in the field. Section III-A formally defines the problem and presents the overview of the model. Section V-B describe the experimental study and the results and VI concludes.

II. RELATED WORK

Time Series Forecasting. To achieve accurate forecasting results, modeling temporal dynamics is an essential component of any time series forecasting model. Traditional methods such as Vector Auto-Regression (VAR) or Auto-Regressive Integrated Moving Average (ARIMA) [31] cannot capture nonlinear patterns and thus fail to capture complex temporal patterns among different time series, resulting in sub-optimal forecasting accuracy [15]. Recent studies show that deep learning methods are able to consistently outperform traditional methods in the context of time series forecasting [17]. Existing studies on deep learning time series forecasting often use three types of models—recurrent neural networks (RNNs) [16]–[20], [32] in the form of LSTM or GRU, and temporal convolutions networks (TCNs) [21]–[23].

Recently attention mechanism such as Transformers [24]–[26], [33], [34] have shown superior

performance when compared with RNN and TCN based models, as they are better at handling long term dependencies. However, they suffer from quadratic memory and runtime overhead w.r.t. to the input sequence length H. To address those limitations, in sliding window attentions, e.g., [35], [36], each timestamp considers a sliding window of length S, covering its past and future neighboring timestamps. As such, each timestamp attends only to the S timestamps in the sliding window. This leads to $\mathcal{O}(H \times S)$ complexity, where H is the length of time series. We instead strive for attentions with linear complexity $\mathcal{O}(H)$. Furthermore, existing attention models, including the sliding window attentions, are still *spatial-temporal agnostic* as they use the same projection matrices irrespective of the time series locations and the time periods.

Categorization of Deep Learning Based Forecasting. We systematically review deep learning studies on time series forecasting from two aspects—spatial awareness vs. temporal awareness. First, spatial agnostic models use the same set of model parameters for time series from different locations, whereas spatial aware models use distinct sets of model parameters for time series from different locations. Second, temporal agnostic models use the same set of model parameters for different time periods, whereas temporal aware models use distinct sets of model parameters for different time periods. The categorization is shown in Table II.

Spatial \ Temporal	Agnostic	Aware
Agnostic	[16], [24], [25], [36]–[40] **[17], [22], [28]–[30], [41]**, **[33], [35]**	**[42]**
Aware	[43], **[18], [44]**	**Ours**

TABLE II: Categorization of Related Studies. The bold references are included in the empirical study, covering the strongest and latest baselines in each category.

Most of the related studies are *spatio-temporal agnostic*, as shown in the top-left corner of Table II. They apply the same model parameters, e.g., projection matrices in self-attention, weight matrices in RNNs, convolution filters in TCNs, for all time series from different locations. By using the same parameter space for all locations such studies implicitly assume that the traffic patterns are similar among locations. Such assumption may not always hold in practice, as demonstrated by a real world example in Figure 1, and this inhibits the model capabilities as the models parameters only learn to capture the "average" traffic dynamics among all locations.

Three existing studies [18], [43], [44] propose *spatial-aware* but still *temporal-agnostic*, RNN based models by using location-specific weight matrices for RNNs. Specifically, [43] generates location-specific weight matrices for RNNs based on additional meta-knowledge, e.g., categories of points of interest around the deployed sensor locations which might not always be available, limiting the applicability of the model, e.g., in the data sets where such additional meta-knowledge is unavailable. [18] learns a pool of candidate weights and choose to combine them differently to obtain location-specific

Variable	Definition
$\mathbf{X}$	Time series data
H	Length of the input time series
U	Length of the predicted time series
N	Number of sensors/time series
$\mathbf{h}^{(i)}$	Output of attention for the i-th sensor
$\boldsymbol{\Theta}_t^{(i)}$	Stochastic latent variable for the i-th sensor
$\mathbf{z}^{(i)}$	Spatial-aware stochastic variable
$\mathbf{z}^{(i)}_t$	Temporal-aware stochastic variable
E_ψ	An encoder network E parameterized by ψ
D_ω	A decoder network D parameterized by ω
W	Number of windows
S	Window size
$\mathbf{P}$	Proxy tensor
$\mathbf{Q}, \mathbf{K}, \mathbf{V}$	Learnable projection matrices in attentions

TABLE III: Table of notations

weight matrices for RNNs. One recent study [44] employs a deterministic memory per location to generate spatial-aware model parameters for time series forecasting, but still fails to enable temporal-aware modeling. In contrast, our proposal proposes to employ a stochastic variable to generate both spatial and temporal aware model parameters. Thus, [44] can be considered as a special case of the paper with the standard deviation of the stochastic variable being always 0 and without temporal-aware parameter generation. We show strong empirical results that our proposal outperforms [18] and [44] (see Table IV). Spatial-aware model parameter generation has also been studied for image recognition and object detection [45]–[47], but fail to capture temporal dynamics and thus cannot be applied directly to enable time series forecasting.

Finally, one existing study [42] is *temporal-aware* but still *spatial-agnostic*. The authors propose to use an additional LSTM which acts as a *meta*-LSTM. The *meta*-LSTM's hidden representation, which varies across time, is then used to generate time-varying model parameters for another LSTM. The empirical study shows that our proposal outperforms [42].

III. PRELIMINARIES

A. Problem Definition

Given a set of N sensors that are deployed in different locations in a spatial network, each sensor produces a multidimensional time series recording F attributes (e.g., traffic speed, traffic flow) across time. Assuming that each time series has a total of T timestamps, we use $\mathbf{X} \in \mathbb{R}^{N \times T \times F}$ to represent the time series from all N sensors. We use $\mathbf{x}^{(i)} \in \mathbb{R}^{T \times F}$ to denote the time series from the i-th sensor, $\mathbf{x}_t \in \mathbb{R}^{N \times F}$ to denote the attributes from all sensors at the t-th timestamp, and $\mathbf{x}_t^{(i)} \in \mathbb{R}^{F}$ to represent the attribute vector from the i-th sensor at the t-th timestamp.

Time series forecasting learns a function $\mathcal{F}$ that, at timestamp t, given the attributes of the past H timestamps from all sensors, predicts the attributes of all sensors in the future U timestamps.

$$\mathcal{F}_\phi(\mathbf{x}_{t-H+1}, ..., \mathbf{x}_{t-1}, \mathbf{x}_t) = (\hat{\mathbf{x}}_{t+1}, \hat{\mathbf{x}}_{t+2}, ..., \hat{\mathbf{x}}_{t+U}), \quad (1)$$

where ϕ denotes the learnable model parameters of the forecasting model and $\hat{\mathbf{x}}_j$ is the prediction at time j. Important notation is shown in Table III.

B. Spatio-temporal Agnostic Attention based Forecasting

We use canonical attention [48], more specifically, self-attention, which is *spatio-temporal agnostic*, as an example to illustrate the needs of *spatio-temporal aware* modeling for traffic forecasting. Figure 2 shows an overview of self-attention based forecasting, where multiple attention layers are stacked and then a predictor, e.g., a neural network, is applied on the output of the last layer to make forecasts $\hat{\mathbf{X}}$.
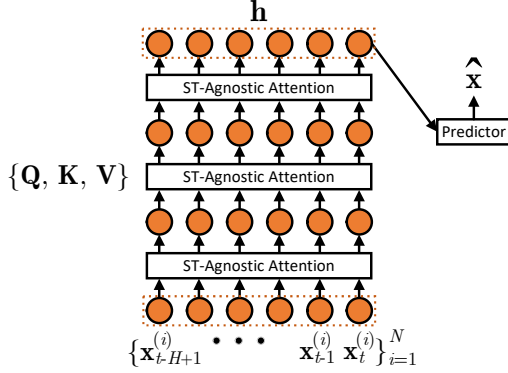

Fig. 2: Spatio-temporal Agnostic Forecasting Model.

In the following, we first cover how to use self-attention to make forecasting for a single time series and multiple time series, respectively, and then illustrate the needs of spatio-temporal aware modeling.

Single Time Series: Consider an input time series with H timestamps, e.g., time series $\mathbf{x}^{(i)} \in \mathbf{R}^{H \times F}$ from the i-th sensor. Self-attention first transforms the time series into a query matrix $\mathbf{x}^{(i)}\mathbf{Q}$, a key matrix $\mathbf{x}^{(i)}\mathbf{K}$ and a value matrix $\mathbf{x}^{(i)}\mathbf{V}$. Here, *projection matrices* $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{F \times d}$ are the learnable model parameters of self-attentions.

The output $\mathbf{h}^{(i)} \in \mathbb{R}^{H \times d}$ of self-attention is represented as a weighted sum of the values in the value matrix, where the weights, a.k.a., attention scores, represent pairwise similarities between any two timestamps and are computed based on the query matrix and the key matrix, as shown in Equation 2.

$$\begin{aligned} \mathbf{h}^{(i)} &= \mathcal{A}tt(\mathbf{x}^{(i)} \mid \mathbf{Q}, \mathbf{K}, \mathbf{V}) \\ &= \sigma\big(\frac{(\mathbf{x}^{(i)}\mathbf{Q})(\mathbf{x}^{(i)}\mathbf{K})^T}{\sqrt{d}}\big)(\mathbf{x}^{(i)}\mathbf{V}), \end{aligned} \quad (2)$$

where σ represents the softmax activation function.

Multiple Time Series: We now consider the time series from all N censors, upon which we can rewrite self-attentions as shown in Equation 3.

$$\begin{aligned} \mathbf{h} &= \mathcal{A}tt(\mathbf{x} \mid \mathbf{Q}, \mathbf{K}, \mathbf{V}) \\ &= \Big\{ \sigma\big(\frac{(\mathbf{x}^{(i)}\mathbf{Q})(\mathbf{x}^{(i)}\mathbf{K})^T}{\sqrt{d}}\big)(\mathbf{x}^{(i)}\mathbf{V}) \Big\}_{i=1}^N, \end{aligned} \quad (3)$$

where $\mathbf{x} \in \mathbb{R}^{N \times H \times F}$ and $\mathbf{h} \in \mathbb{R}^{N \times H \times d}$ correspond to N sensors.

As we aim to capture more complex relationships between different timestamps, it is often useful to stack multiple attention layers to increase the model's representation power [48], and each layer's attention uses its own set of projection matrices. The output of an attention layer is fed as the input of the next attention layer. After multiple layers have been staked, we can take the output of the last layer to feed into a predictor. The predictor can be any type of neural network which is responsible of forecasting future values $\hat{\mathbf{X}}$. Figure 2 illustrates the procedure.

Needs of Spatio-Temporal Aware Modeling: When modeling multiple time series, related studies [24], [25], [29] use the same set of key, query, and value projection matrices $\{\mathbf{Q}, \mathbf{K}, \mathbf{V}\}$, for all N different time series and during different times (cf. the left of Figure 3). This gives rise to a *spatio-temporal agnostic* model that fails to capture distinct patterns from different time series and dynamic patterns that change across time. A spatio-temporal model is called for, where different projection matrices $\{\mathbf{Q}_t^{(i)}, \mathbf{K}_t^{(i)}, \mathbf{V}_t^{(i)}\}$ are used for different time series i and during different times t (cf. the right of Figure 3).

One straightforward solution is to employ a distinct set of projection matrices for each sensor, thus being spatial-aware, and for each time period, thus being temporal-aware. However, this gives rise to a prohibitively large number of parameters to be learned, which leads to high computation time, large memory consumption, slow convergence, and over-fitting issues. Instead, we propose to generate such location-specific and time-varying projection matrices using an encoder-decoder network. The idea is illustrated in Figure 4, where a Spatio-Temporal (ST) Aware Model Parameter Generator generates distinct model parameters $\{\mathbf{Q}_t^{(i)}, \mathbf{K}_t^{(i)}, \mathbf{V}_t^{(i)}\}_{i=1}^N$ for N different sensors' time series and during each time window that ends at time t. This enables spatio-temporal aware attentions without inuring an explosion of model parameters.

IV. METHODOLOGY

A. Spatio-Temporal Aware Modeling

1) Design Considerations: We propose a *data-driven* and *model-agnostic* method to generate location-specific and time-varying parameters, thus being able to turn spatio-temporal agnostic models to spatio-temporal aware models.

First, the method is purely data-driven that only relies on the time series themselves (see the input of the Encoder in Figure 5) and does not rely on any additional information, e.g., points-of-interest around the deployed locations of the sensors, or the knowledge on when morning and afternoon peak hours start at different locations.

Second, the method is not restricted to a specific type of model as the decoder can produce model parameters for different types of models. Figure 5 shows an example decoder that outputs location-specific and time-varying model parameters for attentions, i.e., projection matrices. The above two features ensure smooth transformations of existing, different types
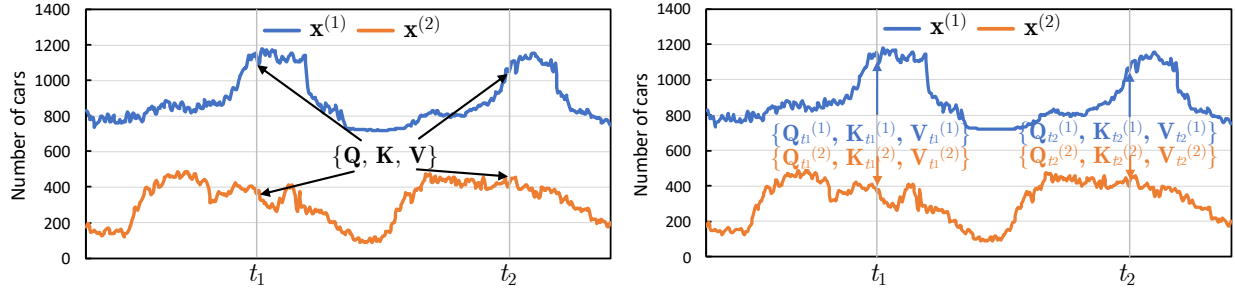
Fig. 3: Spatio-temporal Agnostic Attentions (Left) use the *same* set of projection matrices for different time series across different times. Spatio-temporal Aware Attentions (Right) use *distinct* sets of projection matrices for different time series across different times.
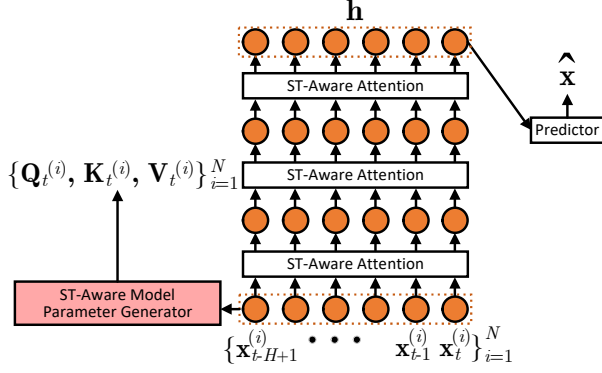

Fig. 4: Spatio-temporal Aware Forecasting Model

Algorithm 1: Efficient ST-Aware Modeling Algorithm

Input: X

1   $\left\{\boldsymbol{\mu}^{(i)}\right\}_{i=1}^{N}, \left\{\boldsymbol{\Sigma}^{(i)}\right\}_{i=1}^{N} \leftarrow$ randomly initialize;

2   $E_{\psi}, D_{\omega}, \left\{\mathbf{P}^{(i)}\right\}_{i=1}^{N} \leftarrow$ randomly initialize;

3   **while** *did not converge* **do**

4   Sample a batch of training data and labels from **X**;

5   Sample $\left\{\mathbf{z}^{(i)}\right\}_{i=1}^{N}$ using Eq. 5;

6   Compute $\left\{\boldsymbol{\mu}_{t}^{(i)}\right\}_{i=1}^{N}, \left\{\boldsymbol{\Sigma}_{t}^{(i)}\right\}_{i=1}^{N}$ using Eq. 6;

7   Sample $\left\{\mathbf{z}_{t}^{(i)}\right\}_{i=1}^{N}$ using Eq. 7;

8      $\left\{\boldsymbol{\Theta}_{t}^{(i)}\right\}_{i=1}^{N} = \left\{\mathbf{z}^{(i)}\right\}_{i=1}^{N} + \left\{\mathbf{z}_{t}^{(i)}\right\}_{i=1}^{N}$;

9   Compute ST-Aware parameters $\left\{\phi_{t}^{(i)}\right\}_{i=1}^{N}$ using Eq. 8;

10   **foreach** *w in W* **do**

11   Update the proxy tensor with information from the previous window using Eq. 14;

12   Compute proxy values $\left\{\mathbf{h}_{w}^{(i)}\right\}_{i=1}^{N}$ using Eq. 10;

13   Compute proxy weights $\left\{\mathbf{A}_{w}^{(i)}\right\}_{i=1}^{N}$ using Eq. 12;

14   Aggregate all proxies into $\left\{\hat{\mathbf{h}}_{w}^{(i)}\right\}_{i=1}^{N}$ using Eq. 13;

15   **end**

16   Optimize parameters using Eq. 20;

17 **end**

of spatio-temporal agnostic models to spatio-temporal aware models without using additional, third party information.

In the following, we use attentions as an example to illustrate how to use the method to turn spatio-temporal agnostic attentions to spatio-temporal aware attentions. In the experiments, in addition to attentions, we also test the method on RNNs to justify the method is model-agnostic.

We proceed to offer an overview of the spatio-temporal aware parameter generation, as shown in Figure 5. We aim to learn a unique stochastic latent variable $\boldsymbol{\Theta}_{t}^{(i)}$ for each location's time series during each time period. More specifically, $\boldsymbol{\Theta}_{t}^{(i)}$ denotes the variable from the i-th time series for a period ending at time t. We design $\boldsymbol{\Theta}_{t}^{(i)}$ as the sum of a spatial-aware variable $\mathbf{z}^{(i)}$ and a temporal adaption variable $\mathbf{z}_{t}^{(i)}$. Then, we generate location-specific and time-varying model parameters using a decoder from $\boldsymbol{\Theta}_{t}^{(i)}$. This facilities us to generate spatio-temporal aware parameters for different models, thus being model-agnostic. In the following, we first cover the learning of stochastic latent variable $\boldsymbol{\Theta}_{t}^{(i)}$ and then model parameter decoding from $\boldsymbol{\Theta}_{t}^{(i)}$.

2) Learning Stochastic Latent Variables $\boldsymbol{\Theta}_{t}^{(i)}$: We design the unique stochastic latent variable $\boldsymbol{\Theta}_{t}^{(i)}$ for the i-th time series during a time period ending at t as the sum of two stochastic latent variables—spatial-aware variable $\mathbf{z}^{(i)}$ and temporal adaption variable $\mathbf{z}_{t}^{(i)}$ (cf. Equation 4).

$$\boldsymbol{\Theta}_{t}^{(i)} = \mathbf{z}^{(i)} + \mathbf{z}_{t}^{(i)} \qquad (4)$$

Here, we expect the spatial-aware variable $\mathbf{z}^{(i)}$ to capture the most general and prominent patterns of the i-th time series and

the temporal adaption variable $\mathbf{z}_{t}^{(i)}$ to accommodate specific variations w.r.t. the most general patterns at different times. We use stochastic variables because they generalize better and have stronger representational power [49] when compared with deterministic variables, which are a special case of stochastic variables with covariance matrix being all 0.

The spatial-aware stochastic latent variable $\mathbf{z}^{(i)}$ is expected to represent the most general and prominent patterns of the i-th time series. It is spatial-aware, as time series from different locations have different variables $\mathbf{z}^{(i)}$, and thus are expected to capture different patterns from different locations.

Since we do not rely on any prior knowledge of the distributions of the latent variables, and many real world processes follow Gaussian distributions [50], we assume they follows a multivariate Gaussian distribution in a k-dimensional space (cf. Equation 5). Moreover, this assumption is well used and accepted in stochastic models such as [49], [51]. In addition,
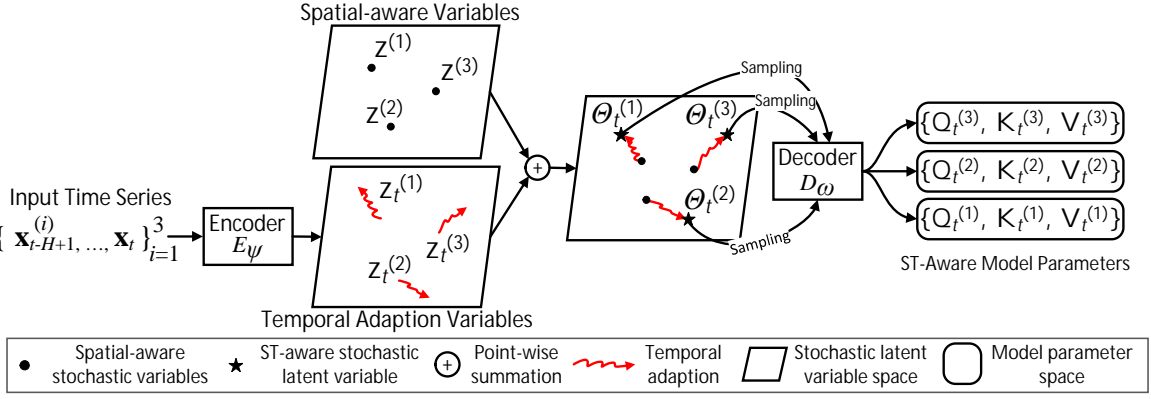
Fig. 5: Spatio-temporal Aware Model Parameters Generation

multivariate Gaussian distributions offer nice properties such as analytical evaluation of the KL divergence in the loss function (cf. Equation 20) and the reparameterization trick for efficient gradient computation. Furthermore, we provide empirical evidence in Table XI which shows that our proposal outperforms its deterministic variant.

$$\mathbf{z}^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}^{(i)}, \boldsymbol{\Sigma}^{(i)}) \tag{5}$$

We let $\mathbf{z}^{(i)}$ be directly learnable in a pure data-driven manner, without using an encoder. This means that the mean $\boldsymbol{\mu}^{(i)}$ and covariance matrix $\boldsymbol{\Sigma}^{(i)}$ are learnable parameters. Alternatively, we may use an encoder to generate $\mathbf{z}^{(i)}$ using location related features, such as POIs around the censor locations [43]. However, such information is often unavailable from time series data itself, but from third parties, which limits the applicability of the method. We thus choose a pure data-drive design.

Next, the temporal adaption variable $\mathbf{z}_t^{(i)}$ represents changes w.r.t. the most prominent patterns captured by $\mathbf{z}^{(i)}$ at a particular time t, which accommodates the pattern changes during at t. Thus, $\mathbf{z}_t^{(i)}$ should be temporal aware as it changes over time. To do so, we produce the temporal adaption variable $\mathbf{z}_t^{(i)}$ conditioned on the most recent H timestamps from the i-th time series. To this end, given $\{\mathbf{x}_{t-H+1}^{(i)}, \ldots, \mathbf{x}_t^{(i)}\}$ consisting of the most recent H timestamps from the i-th time series, inspired by [49] we opt for a variational encoder E_ψ, parameterized by ψ, e.g., a neural network, to generate the stochastic variable $\mathbf{z}_t^{(i)}$, which we assume also follows a multivariate Gaussian distribution (cf. Equations 6 and 7). We opted for a variational encoder to generate model parameters due to its ability to capture the distributions of the input data, which better generalize to cases that were not seen during training.

$$\boldsymbol{\mu}_t^{(i)}, \boldsymbol{\Sigma}_t^{(i)} = E_\psi(\mathbf{x}_{t-H+1}^{(i)}, \ldots, \mathbf{x}_t^{(i)}) \tag{6}$$
$$\mathbf{z}_t^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_t^{(i)}, \boldsymbol{\Sigma}_t^{(i)}) \tag{7}$$

Note that $\boldsymbol{\mu}_t^{(i)}$ and $\boldsymbol{\Sigma}_t^{(i)}$ are the output of the encoder E_ψ, where the encoder parameters in ψ are learnable. This is different from the learning of $\mathbf{z}^{(i)}$ where $\boldsymbol{\mu}^{(i)}$ and $\boldsymbol{\Sigma}^{(i)}$ are directly learnable.

The two stochastic variables $\mathbf{z}^{(i)}$ and $\mathbf{z}_t^{(i)}$ need to be in the same k-dimensional space, as illustrated by the diamond-shape space in Figure 5, such that the sum of them is meaningful, i.e., still in the same space.

3) Decoding to Spatio-Temporal Aware Model Parameters: We use a decoder D_ω parameterized by ω, e.g., a neural network, to decode the learned stochastic latent variable $\boldsymbol{\Theta}_t^{(i)}$ to model parameters $\phi_t^{(i)}$ for a specific type of forecasting model, e.g., attentions, or RNNs, as shown in Equation 8. Since we can not directly backpropagate the gradients through a distribution, which is required to achieve an end to end training, we use the reparameterization trick [49] to circumvent this issue. More specifically, we first obtain samples according to the distribution specified by $\boldsymbol{\Theta}_t^{(i)}$, which are then used as the input to the decoder to generate model parameters (cf. Equation 8).

$$\phi_t^{(i)} = D_\omega(\boldsymbol{\Theta}_t^{(i)}) \tag{8}$$

Figure 5 shows an example where the decoder maps the stochastic latent variables $\boldsymbol{\Theta}_t^{(i)}$ into model parameters for attention, i.e., projection matrices, where we have $\phi_t^{(i)} = \{\mathbf{Q}_t^{(i)}, \mathbf{K}_t^{(i)}, \mathbf{V}_t^{(i)}\}$. When using the obtained location-specific, time-varying projection matrices in attentions, it turns spatio-temporal-agnostic attention into spatio-temporal-aware attention, as shown in Equation 9.

$$\begin{aligned}
\mathbf{h} &= \mathcal{A}tt(\mathbf{x}|\phi_t^{(1)}, \ldots, \phi_t^{(N)}) \\
&= \left\{\sigma\left(\frac{(\mathbf{x}^{(i)}\mathbf{Q}_t^{(i)})(\mathbf{x}^{(i)}\mathbf{K}_t^{(i)})^T}{\sqrt{d}}\right)(\mathbf{x}^{(i)}\mathbf{V}_t^{(i)})\right\}_{i=1}^N
\end{aligned} \tag{9}$$

Figure 4 shows an overview of the forecasting model based on spatio-temporal aware attentions.

Comparing to the naive approach that directly learn projection matrices for each location, the proposed spatio-temporal aware model parameter generation reduces significantly the amount of parameters to be learned. In the naive way where each of the N sensors has its own set of projection matrices.

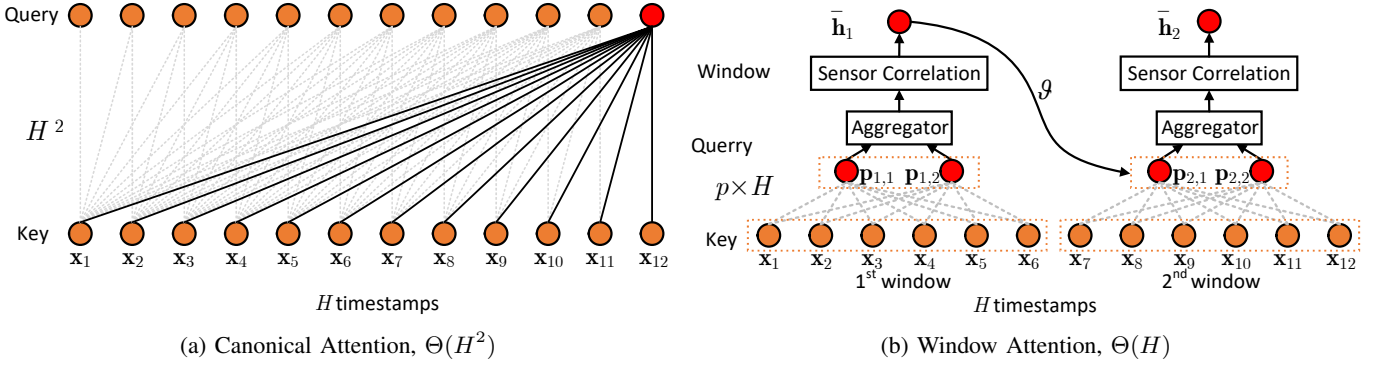(a) Canonical Attention, $\Theta(H^2)$ (b) Window Attention, $\Theta(H)$

Fig. 6: Complexity Comparison between Canonical Attention (CA) vs. Window Attention (WA). In CA, both Key and Query include all H timestamps as the input time series, and each timestamp in Key attends to all timestamps in Query. In WA, we split the input time series into multiple windows (2 windows in this example) and we perform attentions per window. For each window, Query only includes a limited number of p proxies ($p = 2$ in this example) and each timestamp in Key attends to each proxy in Query.

This gives rise to $\mathcal{O}(N \times d^2)$ parameters. Such a high number of parameters require high memory consumption and also lead to over-fitting and poor scalability w.r.t. the number of sensors. In contrast, our proposal requires $\mathcal{O}(N \times k)$ parameters for the spatial stochastic latent variable $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. In addition when using a 2 layer MLP network as the decoder D_ω with m_1 and m_2 neurons then we only need $\mathcal{O}(k \times m_1 + m_1 \times m_2 + m_2 \times d^2)$ as the decoder is shared across all sensors. By doing so we decouple the number of sensors N from d^2 which is the dominant term. Furthermore, now m_1 and m_2 are hyper-parameters which can be controlled such that it offers a trade-off between number of parameters, i.e. memory usage, and efficiency.

B. Efficient Spatio-Temporal Aware Attention

Although attention based models show promising accuracy, a limitation is its quadratic complexity $\mathcal{O}(H^2)$, where H is the size of the input time series. This is because each timestamp in the Key attends to all timestamps in the Query (cf. Figure 6(a)). When the input time series is long, e.g., in settings of long term forecasting, the quadratic complexity becomes a serious barrier.

The spatio-temporal aware parameter generation incurs further computation costs, making its efficiency even worse. To contend with the efficiency challenges, we propose an efficient *Window Attention* (WA) with linear complexity, to ensure competitive overall efficiency, even when using the spatio-temporal aware parameter generation.

We propose to break down the input time series of size H into $W = H/S$ smaller windows along the temporal dimension where S is the window size. We compute attention scores per window; and for each window, we introduce multiple (a small constant, e.g., 2 or 3) learnable proxies which act as the "Query" in the canonical attention. Within a window, each timestamp in the Key only computes a single attention score w.r.t. each proxy in the Query, thus achieving linear complexity.

More specifically, in canonical attention, each of the S timestamps in a window should compute an attention score w.r.t. each timestamp in the windows, thus having $O(S^2)$.

Instead, assuming that we have p proxies in window attentions, each of the S timestamps in the window computes an attention score w.r.t. each proxy, resulting in $O(p \cdot S)$. Since we use a small constant of proxies, i.e., p being a constant, then we have linear complexity $O(S)$ for window attentions.

The idea is illustrated in Figure 6(b) with an example input time series of size $H = 12$ being split into $W = 2$ windows, where the window size is $S = 6$ and each window has $p = 2$ proxies. We proceed to describe the details and the design intuition behind proxies. We assume that for each window there exist p representative temporal patterns, which we use p proxies to capture. The proxies then replace the Query inside canonical self-attention. To this end, we introduce a distinct learnable proxy tensor $\mathbf{P} \in \mathbb{R}^{W \times N \times p \times d}$, such that during each of the W windows, each of the N sensors has a total of p proxies. Each proxy uses a d-dimensional vector, which intends to capture a specific, hidden temporal pattern in the window.

Specifically, when considering the i-th time series and the w-th window, $\mathbf{P}_w^{(i)} \in \mathbb{R}^{p \times d}$ replaces the Query $\mathbf{x}^{(i)} \mathbf{W}_Q \in \mathbb{R}^{H \times d}$ in Equation 3. This serves as a set of p proxies. Then, each time timestamp in window w of the i-th time series needs to compute a single attention score w.r.t. each proxy of the p proxies in $\mathbf{P}_w^{(i)}$. Formally the operation is defined as shown in Equation 10:

$$
\begin{aligned}
\mathbf{h}_w^{(i)} &= \mathcal{A}tt(\mathbf{x}_w^{(i)} \mid \mathbf{P}_w^{(i)}, \mathbf{K}_t^{(i)}, \mathbf{V}_t^{(i)}) \\
&= \left\{ \sigma\left(\frac{\mathbf{P}_{w,j}^{(i)} (\mathbf{x}_w^{(i)} \mathbf{K}_t^{(i)})^T}{\sqrt{d}} \right)(\mathbf{x}_w^{(i)} \mathbf{V}_t^{(i)}) \right\}_{j=1}^{p},
\end{aligned} \tag{10}
$$

where $\mathbf{x}_w^{(i)} = (\mathbf{x}_{w*S+1}^{(i)}, \mathbf{x}_{w*S+2}^{(i)}, ..., \mathbf{x}_{(w+1)*S}^{(i)})$ and the output of window w for time series i is $\mathbf{h}_w^{(i)} \in \mathbb{R}^{p \times d}$. To get the output of the layer we need to apply Equation 11 for each window w:

$$
\mathbf{h}^{(i)} = \left\{ \mathcal{A}tt(\mathbf{x}_w^{(i)} | \mathbf{P}_w^{(i)}, \mathbf{K}^{(i)}, \mathbf{V}^{(i)}) \right\}_{w=0}^{W-1}, \tag{11}
$$

where $\mathbf{h}^{(i)} \in \mathbb{R}^{W \times p \times d}$. Note that by applying the proxies, we reduce the complexity of canonical self-attention on an input

time series of size H from quadratic $\mathcal{O}(H^2)$ to $\mathcal{O}(W \cdot p \cdot S)$. Since the number of proxies p is a constant and since $W \cdot S = H$, the asymptotic complexity becomes linear, i.e., $\mathcal{O}(H)$.

To further reduce the complexity for the upcoming layers, we propose an aggregation function responsible of aggregating all the p proxies within a window w, i.e., $\mathbf{h}_w^{(i)} = \{\mathbf{h}_{w,j}^{(i)}\}_{j=1}^p$, into a single, shared hidden representation $\hat{\mathbf{h}}_w^{(i)}$. The resulting $\hat{\mathbf{h}}_w^{(i)}$ is shrinked by a factor of p, thus giving rise to less attention scores computation at the next layer. The idea is illustrated in Figure 7.
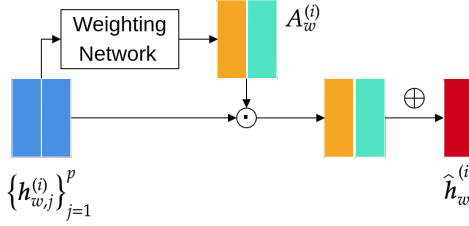

Fig. 7: Illustration of Proxy Aggregator

First, we use a weighting network, e.g., 2-layer neural network as shown in Equation 12, to weigh each proxy.

$$\mathbf{A}_w^{(i)} = f_2(\mathbf{W}_2(f_1(\mathbf{W}_1 \mathbf{h}_w^{(i)}))) \qquad (12)$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable parameters which are shared among different windows. f_1 is a non-linear activation, which is set to $tanh$ in our experiments, and f_2 represents the $Sigmoid$ activation function which ensures the weights are bounded in range $[0, 1]$, controlling the information flow. Next, after we have computed the proxy weight scores $\mathbf{A}_w^{(i)} \in \mathbb{R}^{p \times d}$ we can calculate the final window representation as follows:

$$\hat{\mathbf{h}}_w^{(i)} = \sum_{j=1}^p \mathbf{A}_{w,j}^{(i)} \odot \mathbf{h}_{w,j}^{(i)} \qquad (13)$$

where $\odot$ represents the point-wise multiplication. We first scale each proxy $\mathbf{h}_p^{(i)}$ with its respective weight coefficient $\mathbf{A}_p^{(i)}$ and then sum all the proxies within the same window.

The reduced complexity comes with a price that the temporal receptive filed of each timestamp in window attention is reduced from H, as in the canonical attention, to S, i.e., the window size S. This makes it impossible to capture relationships among different windows and also long-term dependencies, thus adversely affecting accuracy. To compensate for reduced temporal receptive field, we connect the output of a window to the next window proxies such that the temporal information flow is maintained. More specifically, we utilize a function ϑ, e.g., a neural network, to fuse the output of the previous window $\hat{\mathbf{h}}_{w-1}^{(i)}$ with all p learnable proxies $\{\mathbf{P}_{w,j}^{(i)}\}_{j=1}^p$ in the current window before the window attention computation (see ϑ in Figure 6(b)).

$$\mathbf{P}_{w,j}^{(i)} = \vartheta(\hat{\mathbf{h}}_{w-1}^{(i)} \,||\, \mathbf{P}_{w,j}^{(i)}) \quad \forall j \in [1, p] \qquad (14)$$

C. Sensor Correlation Attention

Traffic time series from different sensors often show correlations—traffic from one sensor is often influenced by traffic from nearby sensors, thus the traffic information captured by the sensors show correlations. In order to accurately predict the future, we need a way to model such correlations, which we refer to as *sensor correlations*. We proceed to elaborate how we can utilize the same attention mechanism introduced already to capture the interactions between different sensors within a window w. After we aggregate the proxies, the output of a window attention is $\hat{\mathbf{h}}_w \in \mathbb{R}^{N \times d}$, where the window information is summarized into a fixed representation for the i-th sensor, where $1 \leq i \leq N$. To capture how one sensor i if affected by another sensor j, we use a normalized embedded Gaussian function:

$$\mathbf{B}(\hat{\mathbf{h}}_w^{(i)}, \hat{\mathbf{h}}_w^{(j)}) = \frac{e^{\theta_1(\hat{\mathbf{h}}_w^{(i)})^\mathsf{T} \theta_2(\hat{\mathbf{h}}_w^{(j)})}}{\sum_{j=1}^N e^{\theta_1(\hat{\mathbf{h}}_w^{(i)})^\mathsf{T} \theta_2(\hat{\mathbf{h}}_w^{(j)})}}, \qquad (15)$$

where θ_1 and θ_2 are two embedding functions to perform a linear transformation on the data between sensors i and j. The numerator of Equation 15 calculates the similarity between the source sensor i and the target sensor j, while the denominator ensures that the final attention scores for a specific target sensor is normalised. Next, we can get a new updated representation of the i-th sensor $\bar{\mathbf{h}}^{(i)} \in \mathbb{R}^{N \times d}$ as shown in Equation 16:

$$\bar{\mathbf{h}}^{(i)} = \sum_{j=1}^N \mathbf{B}(\hat{\mathbf{h}}_w^{(i)}, \hat{\mathbf{h}}_w^{(j)}) \odot \hat{\mathbf{h}}^{(j)}, \qquad (16)$$

where $\odot$ represents points-wise multiplication. Similar to temporal attention, we assume that a single set of linear transformation θ_1 and θ_2 might not be sufficient to describe all the interactions between different sensors, and thus we can use the model parameters generation process already introduced in Section IV to generate a distinct set of transformation matrices for each sensor.

D. Full Model

When using attentions, multiple layers of attentions are often stacked together to improve accuracy. Figure 8 shows the full model, which consists of multiple layers of Spatio-Temporal Aware Window Attention (ST-WA), where location-specific and time-varying projection matrices are used. Each window attention layer has its own set of learnable proxy tensor $\mathbf{P}$. The output of the l-th layer is derived by concatenating the output of each window:

$$\mathbf{O}_l = [\bar{\mathbf{h}}_1, \bar{\mathbf{h}}_2, ..., \bar{\mathbf{h}}_W], \qquad (17)$$

where $\mathbf{O}_l \in \mathbb{R}^{W \times N \times d}$. To ease gradient propagation we create gradient shortcuts by connecting each layer output directly to the predictor. Since the dimension of the input is exponentially decreased at each layer we use a skip-connection in the form

of 1 layer neural network to ensure consistent output shape between each layer as shown in Equation 18.

$$\mathbf{O} = \sum_{l=1}^{L} \mathbf{W}_l(\mathbf{O}_l), \tag{18}$$

where L represents the total number of layers, and $\mathbf{W}_l$ represents the learnable parameters of the skip connection at layer l. Finally we employ a predictor with a 2 layer neural network which is responsible of mapping the hidden representation $\mathbf{O}$ to future predictions $\hat{\mathbf{X}}$ as shown in Equation 19.

$$\hat{\mathbf{X}} = \mathbf{W}_4(ReLU(\mathbf{W}_3(\mathbf{O}))), \tag{19}$$

where $ReLU$ represents the rectified linear unit activation function, $\mathbf{W}_3$ and $\mathbf{W}_4$ are learnable weight matrices.
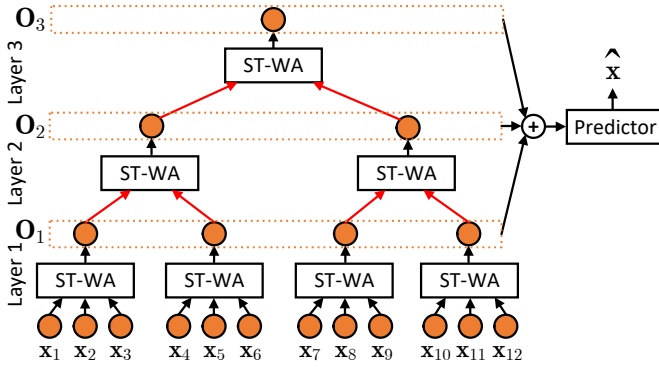

Fig. 8: Full Model

Complexity Analysis When using attention, multiple layers of attention are often stacked together to improve accuracy. In canonical self-attentions, each attention layer has the same input size, e.g., the input time series size H. Thus, when having L layers, the complexity is $\Theta(LH^2)$.

When using window attention, the size of $(l+1)$-th layer is only $\frac{1}{S_l}$ of the size of the l-th layer, where S_l is the window size of the l-th layer. Consider the example shown in Figure 8 with 3 layers of window attention (WA). The size of the input time series is $H = 12$. The window size in the first layer S_1 is 3, making the input of the 2-nd layer WA to be $\frac{1}{3} \cdot 12 = 4$. As the 2-nd layer's window size S_2 is 2, the input of the 3-rd layer WA has an input size of $\frac{1}{2} \cdot 4 = 2$. Thus, the input size of the L-th layer is at most $\frac{H}{\hat{S}^{L-1}}$, where $\hat{S} = \min_{1 \le i \le L} S_i$ is the minimum window size of all layers. Since $\sum_{i=1}^{L} \frac{H}{\hat{S}^{i-1}} < \frac{\hat{S}}{\hat{S}-1} \cdot H$, the complexity of L stacks of WA is still linear $\mathcal{O}(H)$.

E. Loss Function and Optimization

We train the proposed spatio-temporal aware parameters generation by optimizing the loss function shown in Equa-

tion 20 in an end-to-end fashion.

$$\underset{\psi,\omega,\{\boldsymbol{\mu}_i\},\{\boldsymbol{\Sigma}_i\}}{\operatorname{argmin}} \sum_t \mathcal{H}(\mathbf{X}_{t+1,\ldots,t+U}, \hat{\mathbf{X}}_{t+1,\ldots,t+U})$$
$$+ \alpha D_{KL}[\Theta_t \,||\, \hat{p}] \tag{20}$$

The first term measures the discrepancies between the predictions $\hat{\mathbf{X}}_{t+1,\ldots,t+U}$ and the ground truth values $\mathbf{X}_{t+1,\ldots,t+U}$. We use Huber loss to measure the discrepancies in our experiments, since it is less sensitive to outliers in the training data [52].

$$\mathcal{H}(\mathbf{X}_U, \hat{\mathbf{X}}_U) = \begin{cases} \frac{1}{2}(\mathbf{X}_U - \hat{\mathbf{X}})_U^2 & |\mathbf{X}_U - \hat{\mathbf{X}}_U| \le \delta \\ \delta(|\mathbf{X}_U - \hat{\mathbf{X}}_U| - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \tag{21}$$

Here, δ is a threshold parameter that controls the range of squared error loss. The second term is a regularizer w.r.t. a prior distribution $\hat{p} = \mathcal{N}(0, \mathbf{I})$.

Different from traditional learning settings, where we directly optimize the model parameters, e.g., projection matrices in attention. In our setting, the model parameters are generated from a stochastic latent variable by a decoder. Thus, the learning goal is to optimize the parameters that generate the model parameters. More specifically, the decoder D_ω is parameterized by ω; the stochastic hidden variable Θ_t is the sum of the location specific stochastic variable decided by the learnable parameters $\{\boldsymbol{\mu}_i\}_{i=1}^N$ and $\{\boldsymbol{\Sigma}_i\}_{i=1}^N$; and the stochastic temporal adaption that is returned by the encoder E_ψ parameterized by ψ. Thus, minimizing the loss function enables the identification of the optimal parameters ω, ψ, $\{\boldsymbol{\mu}_i\}_{i=1}^N$ and $\{\boldsymbol{\Sigma}_i\}_{i=1}^N$. When learning the stochastic latent variable Θ_t, in order to enforce our assumption that it follows a multivariate Gaussian distribution, we make the learned posterior distribution close to a prior distribution $\hat{p} = \mathcal{N}(0, \mathbf{I})$. Here, D_{KL} represents the Kullback–Leibler divergence and α is a penalty term to control the contribution of the regularization term. We empirically study the impact of the regularizer and reported the results in the Section V-B.

V. EXPERIMENTAL STUDIES

A. Experimental Setup

Data Sets: We use four commonly used, public traffic time series data sets [18], [28], [30]. The data has been collected by California Transportation Agencies Performance Measurement System (PEMS, http://pems.dot.ca.gov/) and is released by [30]. Every 5 minutes, a time series has $F = 1$ attribute, representing the average traffic flow. Thus, no personally identifiable information, e.g., information on individual vehicles, is revealed. To facilitate fair comparisons, we follow existing studies [18], [29] by splitting the data sets in chronological order with 60% for training, 20% for validation, and 20% for testing. The number of time series N and the duration of each dataset is shown in Table IV.

Forecasting setting: We consider a commonly used forecasting setup as the default setup [17], [22], [30], where given

Baseline methods		ST-agnostic								S-aware		T-aware	ST-aware
Dataset	Metric	LongFormer	DCRNN	STGCN	STG2Seq	GWN	STSGCN	ASTGNN	STFGNN	EnhanceNet	AGCRN	meta-LSTM	ST-WA
PEMS03	MAE	17.50	18.18	17.49	19.03	19.85	17.48	**15.07**	16.77	16.05	16.06	19.89	15.17
N=358	MAPE	16.80	18.91	17.15	21.55	19.31	16.78	**15.80**	16.30	15.83	15.85	20.38	15.83
T=3mo	RMSE	30.24	30.31	30.12	29.73	32.94	29.21	26.88	28.34	28.33	28.49	33.71	**26.63**
PEMS04	MAE	23.83	24.70	22.70	25.20	25.45	21.19	19.26	19.83	20.44	19.83	25.37	**19.06**
N=308	MAPE	15.57	17.12	14.59	18.77	17.29	13.90	12.65	13.02	13.58	12.97	17.09	**12.52**
T=2mo	RMSE	37.19	38.12	35.55	38.48	39.70	33.65	31.16	31.88	32.37	32.26	39.90	**31.02**
PEMS07	MAE	26.80	25.30	25.38	32.77	26.85	24.26	22.23	22.07	21.87	21.29	27.02	**20.74**
N=883	MAPE	12.11	11.66	11.08	20.16	12.12	10.21	9.25	9.21	9.13	8.97	12.32	**8.77**
T=4mo	RMSE	42.95	38.58	38.78	47.16	42.78	39.03	35.95	35.80	35.57	35.12	43.00	**34.05**
PEMS08	MAE	18.52	17.86	18.02	20.17	19.13	17.13	15.98	16.64	16.33	15.95	19.99	**15.41**
N=170	MAPE	13.66	11.45	11.40	17.32	12.68	10.96	9.97	10.60	10.39	10.09	12.51	**9.94**
T=2mo	RMSE	28.68	27.83	27.83	30.71	31.05	26.80	25.67	26.22	25.46	25.22	31.65	**24.62**

TABLE IV: Overall Accuracy, $H = 12$, $U = 12$

the previous $H = 12$ timestamps (1 hour), we predict the following $U = 12$ timestamps (1 hour). We consider two additional setups. First, we increase the historical timestamps to $H = 36$ (3 hours) and $H = 120$ (10 hours) while keeping $U = 12$ to study the impact on long term dependency. Second, we increase the historical timestamps to $H = 72$ (6 hours) and the forecasting horizon to $U = 72$ (6 hours) to quantify the scalability of the methods.

Baselines: To evaluate the performance of our proposal, we compare with widely used baselines and state-of-the-art models for traffic time series forecasting. In particular, this includes ST-agnostic models (`LongFormer`, `DCRNN`, `STGCN`, `STG2Seq`, `GWN`, `STSGCN`, `ASTGNN`, `STFGNN`), S-aware models (`EnhanceNet`, `AGCRN`), and T-awre models (`meta-LSTM`).

- `LongFormer` [35] a Transformer like architecture which employs sliding window attention.
- `DCRNN` [17] employs `GRU` to model temporal dependencies (*TD*) and diffusion graph convolution to model sensor correlations (*SC*).
- `STGCN` [29] uses 2D convolution to model *TD* and graph convolution to model *SC*.
- `STG2Seq` [41] uses gated residual with attentions to model *TD* and graph convolution to model *SC*.
- `GWN` [22] Graph WaveNet uses dilated causal convolutions to model *TD* and graph convolution for *SC*.
- `STSGCN` [30] a state-of-the-art method which jointly captures *TD* and *SC* using graph convolution.
- `ASTGNN` [33] an encoder-decoder network which incorporates local context into self-attention.
- `STFGNN` [28] relies on gated convolutions to jointly capture *TD* and *SC*.
- `EnhanceNet` [44] A state-of-the-art spatial-aware method that enhances existing `RNN` and `TCN` based models for *TD* and graph convolution for *SC*.
- `AGCRN` [18] A state-of-the-art spatial-aware method that uses `RNN` for modeling *TD* and graph convolution for *SC*, achieving the best accuracy so far.
- `meta-LSTM` [42] a temporal-aware method that uses `LSTM` to capture *TD*, while *SC* is not captured.

We do not include another spatial-aware model `ST-MetaNet` [43] as it requires POI information. However, no POI information is available along with the data sets and all other baselines do not rely on POI information.

Evaluation Metrics: For accuracy, we follow existing literature [17], [22] to report mean absolute error (*MAE*), root mean square error (*RMSE*), and mean absolute percentage error (*MAPE*). We also report runtime and memory usage.

Implementation Details and Hyperparameter Settings: We train our model using Adam optimizer with a fixed learning rate of 0.001 and with a batch size of 64. The total number of epochs is set to 200 and we use early stopping with a patience of 15. We tune the hyper-parameters on the validation data by grid search. The number of layers is chosen from $L \in \{1, 2, 3\}$, window size $w \in \{2, 3, 6\}$, number of proxies $p \in \{1, 2, 3\}$, hidden representation $d \in \{16, 32\}$, stochastic latent variable size $k \in \{4, 8, 16, 32\}$.

For $H = 12$ we have a default setting in which we stack 3 layers, with $p = 1$ and we set the window size S in the 1st, 2nd, and 3rd layers to be 3, 2, and 2, respectively. For $H = 72$ we use 3 layers with $p = 2$ and $S = 6$ across all layers. We set the hidden representation $d = 32$. We initialize each sensor's stochastic latent variable $\mathbf{z}^{(i)}$ as a $k = 16$ dimensional Gaussian distribution using randomly initialized parameters $\mu^{(i)}$ and $\Sigma^{(i)}$. We use a 3 layer fully connected network with 32 neurons and $ReLU$ as the activation for the encoder E_ψ to generate a 16-dimensional Gaussian distribution $\mathbf{z}_t^{(i)}$ parameterized by $\mu_t^{(i)}$ and $\Sigma_t^{(i)}$. We enforce $\Sigma^{(i)}$ and $\Sigma_t^{(i)}$ to be diagonal matrices. We use a 3 layer fully-connected network with 16, 32, and 5 neurons and $ReLU$ as activation functions for the first 2 layers as the decoder D_ω. For the predictor we use 2 fully connected layers, each with 512 neurons, and $ReLU$ as activation function. When computing the attentions, to learn different types of relationships, we utilize multi-head attention with a total of 8 heads.

We implement `ST-WA` and other baselines on Python 3.7.3 using PyTorch 1.7.0. The experiments are conducted on a computer node on an internal cloud, running Ubuntu 16.04.6 LTS, with one Intel(R) Xeon(R) CPU @ 2.50GHz with one Tesla V100 GPU card. The code is publicly available on GitHub at https://github.com/razvanc92/ST-WA.

B. Experimental Results

Overall Accuracy. Table IV shows overall accuracy on the default forecasting setup in which $H = 12$ and $U = 12$. First,

	H = 12				H = 36				H = 120			
	STFGNN	EnhanceNet	AGCRN	ST-WA	STFGNN	EnhanceNet	AGCRN	ST-WA	STFGNN	EnhanceNet	AGCRN	ST-WA
MAE	19.83	20.44	19.38	**19.06**	20.93	20.26	19.69	**18.90**	19.36	19.85	19.87	**18.90**
MAPE	13.02	13.58	12.89	**12.52**	13.74	13.29	12.91	**12.43**	13.20	13.53	13.59	**12.92**
RMSE	31.88	32.37	31.29	**31.02**	32.90	32.09	32.27	**30.92**	30.94	31.63	31.65	**30.69**

TABLE V: Impact of H, **PEMS04**

Baseline methods		STFGNN	EnhanceNet	AGCRN	ST-WA
Dataset	Metric				
PEMS03 N=358	MAE	24.08	23.42	22.17	**20.87**
	MAPE	23.89	23.69	23.11	**22.33**
	RMSE	39.91	38.99	37.10	**35.53**
PEMS04 N=307	MAE	25.79	28.72	57.41	**23.54**
	MAPE	18.67	20.68	47.40	**16.52**
	RMSE	39.87	45.20	86.02	**36.87**
PEMS07 N=883	MAE			49.64	**25.04**
	MAPE	OOM	OOM	23.43	**10.85**
	RMSE			73.50	**38.62**
PEMS08 N=170	MAE	22.74	37.56	46.50	**19.27**
	MAPE	15.59	23.31	19.05	**13.30**
	RMSE	35.95	64.51	74.78	**30.69**

TABLE VI: Overall Accuracy, $H = 72$, $U = 72$.

Baseline methods		GRU	GRU+S	GRU+ST	ATT	ATT+S	ATT+ST
Dataset	Metric						
PEMS03	MAE	19.97	18.73	**18.68**	20.03	18.08	**17.87**
	MAPE	19.62	20.67	**17.61**	20.01	17.10	**17.00**
	RMSE	32.77	31.22	**31.08**	32.71	30.15	**29.86**
PEMS04	MAE	26.02	24.48	**23.95**	25.84	23.33	**23.09**
	MAPE	17.23	17.11	**15.59**	17.18	16.13	**15.57**
	RMSE	40.11	37.68	**37.18**	39.99	36.63	**36.61**
PEMS07	MAE	27.60	26.08	**25.93**	27.39	24.57	**24.47**
	MAPE	12.41	11.55	**11.34**	12.05	10.99	**10.72**
	RMSE	43.28	40.70	**40.68**	43.41	39.58	**39.55**
PEMS08	MAE	20.75	19.19	**19.02**	20.54	17.95	**17.76**
	MAPE	13.20	12.44	**12.40**	14.11	11.92	**11.85**
	RMSE	32.15	29.75	**29.41**	32.01	28.45	**28.30**

TABLE VII: Overall Accuracy of Enhanced versions of `RNN` and `ATT`, $H = 12$, $U = 12$

we observe that temporal-aware model `meta-LSTM` performs the worst. This is due to that it is the only baseline which does not explicitly model the sensor correlations. Next, we observe that the spatial-aware methods, i.e. `EnhanceNet` and `AGCRN`, outperform most of the ST-agnostic baselines, confirming our analysis that using a shared parameter space for time series from different locations is not optimal when modeling spatial-temporal dynamics. Furthermore our proposed method `ST-WA` achieves the best accuracy on 10 out of 12 metrics over four data sets, which justifies our design choices and highlights the importance of explicitly modeling spatio-temporal model parameters.

Next, to study the ability of forecasting long into the future we move away from the classic 1h to 1h setting and increase both H and U to 72 which represents 6 hours. The results are shown in Table VI. We only compare `ST-WA` with `AGCRN`, `EnhanceNet` and `STFGNN`, which are the top-3 baselines from the previous experiment. We can observe that our proposed method significantly outperforms the baselines. Furthermore we observe that `EnhanceNet` and `STFGNN` run out of memory (shown as OOM in Table VI) when the number of sensors N is big, e.g., 883 in the case of **PEMS07**. In contrast, `ST-WA` still works well, suggesting its high memory efficiency.

Model-Agnostic ST-Aware Model Parameter Generation. In this experiment, we demonstrate that the proposed ST-Aware Model Parameter Generation is model-agnostic, which is able to be applied to different types of forecasting models and consistently enhance their accuracy. We consider two types of forecasting models—`GRU` and self-attention based Transformer `ATT`. We first compare the base `GRU` and `ATT` with their enhanced versions `+S` denotes *spatial-aware* and `+ST` stands for *spatial-temporal aware* variants. Table VII shows the accuracy. We can see that `+S` offers clear improvements and `+ST` further improves. This proves that our proposal is model-agnostic and it can easily be integrated into existing

methods to further improve their accuracy.

Ablation Study. We perform an ablation study on the PEMS04 dataset by removing different components from the full model `ST-WA`. First, `S-WA` is a model by removing the $\mathbf{z}_t^{(i)}$ from the full model. This corresponds to a model where weight generation only generates location-specific but not time-varying model parameters. Second, `WA` is stacked Window Attention without ST-aware projection matrix generation. Third, `WA-1` is Window Attention, by removing the stacked structures from `WA`, i.e., a single layer of window attention. Third, we consider `SA` that employs canonical self-attention, i.e., Transformers.

	SA	WA-1	WA	S-WA	ST-WA
MAE	23.31	19.56	19.38	19.25	**19.06**
MAPE	15.69	12.96	12.89	12.81	**12.52**
RMSE	36.76	31.55	31.29	31.37	**31.02**
Memory	23.94	**3.03**	4.44	8.02	8.14
Training	44.22	**7.88**	13.25	21.62	21.93
# Para	393k	**164k**	490k	392k	396k

TABLE VIII: Ablation study on **PEMS04**

Table VIII shows the accuracy along with training time (seconds per epoch), memory usage (GB), and number of parameters to be learned. First, when comparing `SA` with `WA-1`, `WA-1` is more than 3x faster and requires 5x less memory, suggesting the window attention is accurate and efficient, compared to canonical self-attention. `WA` further improves accuracy over `WA-1`, suggesting that the multi-layer hierarchical structure is effective. `S-WA` and `ST-WA` achieve a clear accuracy improvement, and `ST-WA` achieves the highest accuracy, highlighting the need of having both spatial and temporal aware model parameters. For run-time, both models take longer time, while sill being comparable with other baselines (c.f. Figure 10). We conclude that, when training run-time is critical, we recommend the use of `WA-1` or `WA`. Otherwise, `ST-WA` is the best choice.

Impact of Long Historical Window H: We systematically increase H from 12 to 36 and 120. The results are shown in Table V. `ST-WA` shows improvements with longer H while

(a) 2D Points of $\phi_t^{(i)}$ vs. Input Time Series $\mathbf{x}_t^{(i)}$ (b) 2D Points of $\mathbf{z}^{(i)}$ (c) Sensor Locations.
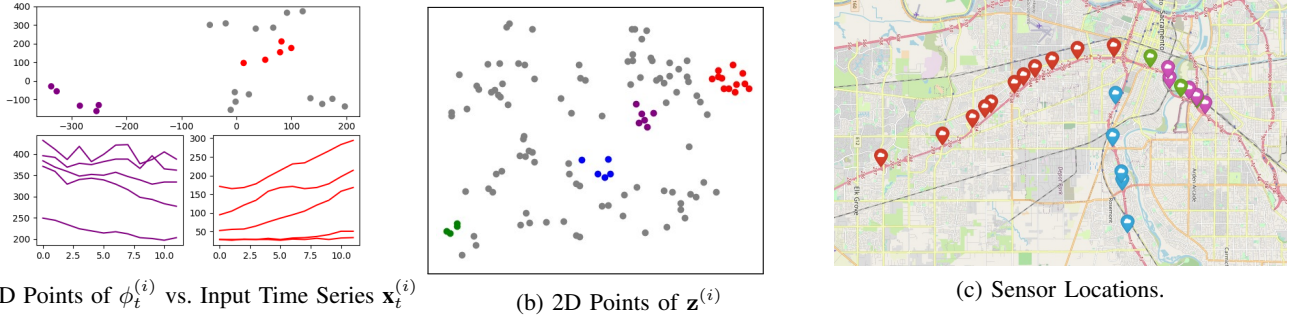
Fig. 9: Justifying Design Choices with Visualization.

the other baselines does not improve significantly, some even loosing accuracy. This suggests that the proposed model is able to better capture long term dependencies while other baselines struggle to do so.

Impact of Window Size S: To investigate the effect of the window size S, we run multiple experiments in which we change the window size and the total number of layers of ST-WA. The results are shown in Table IX. We observe that, when using the window size equal to the length of the sequence $S = H = 12$, it performs the worst. This is understandable, as it only uses 1 layer, the total number of parameters is significantly smaller resulting in lower representation power. Next, we can see that when using 3 layers, there are small variations among different window size settings, suggesting the proposed method is insensitive to window sizes.

	3 layers			2 layers		1 layer
	S=3,2,2	S=2,3,2	S=2,2,3	S=4,3	S=6,2	S=12
MAE	**19.06**	19.10	19.13	19.62	19.69	22.05
MAPE	**12.52**	12.62	12.58	13.13	12.86	14.99
RMSE	**31.02**	31.05	31.15	31.51	31.81	35.35

TABLE IX: Effect of Window Sizes, **PEMS04**

Impact of the Regularizer: To further investigate the effect of the KL based regularization term, we perform an experiment in which we compare the full model ST-WA with and without using the regularization term in the loss function. We can see from Table X that removing the regularization term results in a clear accuracy loss, which justifies our design choice.

	With	Without
MAE	**19.06**	19.23
MAPE	**12.52**	12.62
RMSE	**31.02**	31.37

TABLE X: Effect of the Regularization Term, **PEMS04**

Impact of Stochastic Latent Variables: We perform an additional ablation study in which we compare the proposed method ST-WA with a deterministic version in which we replace the stochastic latent variables, $\mathbf{z}^{(i)}$ and $\mathbf{z}_t^{(i)}$, with deterministic latent variables, i.e., 16-dimensional vectors. In addition, we remove the KL based regularization from the loss function, as it is no longer needed for deterministic latent variables. We can see from Table XI that the stochastic version constantly outperforms the deterministic version, which justi-

fies the effectiveness of our design choice on using stochastic variables.

	MAE	MAPE	RMSE
ST-WA	**19.06**	**12.52**	**31.02**
Deterministic ST-WA	19.32	12.72	31.41

TABLE XI: Effect of stochastic latent variables, **PEMS04**

Impact of Stochastic Latent Variables Size k: We study the impact of the size of stochastic latent variable, i.e., k. Recall that the stochastic latent variables follow a multivariate Gaussian distribution in a k-dimensional space. We perform an additional experiment in which we vary k, i.e., the size of $\mathbf{z}^{(i)}$ and $\mathbf{z}_t^{(i)}$ among $\{4, 8, 16, 32\}$. The results are shown in Table XII. We can observe that when the latent variable size is small, i.e. 4, there is a significant drop in performance, as 4-dimensional Gaussian distributions might not be sufficient to fully capture the dynamics of the traffic conditions. When the size is too large, i.e., 32, we see again a big drop in performance which is due to over-fitting.

k	MAE	MAPE	RMSE
4	19.42	12.62	31.02
8	19.37	12.58	31.15
16	**19.06**	**12.52**	**31.02**
32	19.42	12.96	31.09

TABLE XII: Effect of Latent Stochastic Variables Size k, **PEMS04**

Impact of Number of Proxies p: To study the impact of number of proxies p we have conducted an additional ablation study in which we vary p from 1, 2, to 3. The results are shown in Table XIII. We can observe that the bigger the number of proxies is, the better the accuracy is, but this accuracy improvement comes by compromising on training efficiency (s/epoch) and memory usage on the parameters.

p	MAE	MAPE	RMSE	Training	# Para
1	23.93	16.48	37.44	**69.15**	**606k**
2	23.54	16.52	36.87	102.93	764k
3	**23.48**	**16.50**	**36.70**	128.25	922k

TABLE XIII: Effect of number of Proxies, **PEMS04**

Impact of Aggregation Function: To study the impact of the aggregator function, we ran an additional experiment in which we compare with a mean aggregator, where each proxy within a window is weighted equally. We can observe from Table XIV that when using a simple mean aggregator the

accuracy drops significantly, which justify our design choice (cf. Equation 13).

	MAE	MAPE	RMSE
Mean Aggregator	24.65	17.21	38.41
Our Aggregator	**23.54**	**16.52**	**36.87**

TABLE XIV: Effect of different Aggregation Functions, **PEMS04**

Run-time: To study the efficiency of our proposed method we have reported the training run-time (seconds/epoch) by systematically increasing the historical window H from 12 to 36 and 120, corresponding to 1, 3, and 12 hours respectively. The results are shown in Figure 10. We can observe that for short historical windows `STFGNN`, `AGCRN` and `ST-WA` have similar run-time, while `EnhanceNet` takes significantly longer to train. In addition we can see an exponential growth in run-time w.r.t the sequence length for all baselines. In contrast `ST-WA` is significantly more efficient which increases its usability to applications where processing long-sequence data is required.
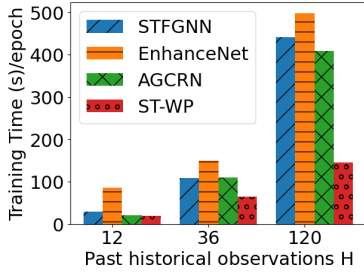

Fig. 10: Runtime, **PEMS04**

C. Visualization of Learned Stochastic Variables

To investigate if the learned stochastic variables are able to capture different patterns during different time windows, we use `t-SNE` [53] to embed the projection matrices $\phi_t^{(i)} = \{\mathbf{Q}t^{(i)}, \mathbf{K}_t^{(i)}, \mathbf{V}_t^{(i)}\}$ during different time windows from a specific time series to 2D points, because the projection matrices are generated from the learned stochastic variables. From Figure 9(a), we observe that the 2D points spread over the space, indicating that different projection matrices are utilized to capture different patterns during different time windows. Next, we have selected two clusters of the 2D points and show their input time series in the corresponding windows. We observe that the red and purple point clusters specialise in identifying up/down trends, respectively. This provides evidence that the proposal is able to capture well temporal dynamics.

We proceed to investigate if the learned location-specific stochastic latent variable $\mathbf{z}^{(i)}$ capture distinct, and the most prominent patterns of time series from different locations. We use `t-SNE` [53] to compress each time series's $\mathbf{z}^{(i)}$ to a 2D space, which is show in Figure 9(b). We observe that the points are spread over the space, indicating that time series from different locations have their unique patterns. We use four different colors to highlight four clusters of the 2D points. Then, we show their physical locations of the deployed sensors

on a map in Figure 9(c). We observe that each cluster contains sensors deployed along the same road. This is intuitive since traffic patterns are expected to follow similar patterns along the same street. In addition, we observe that green and purple sensors are close in Figure 9(c) but their 2D points are far in Figure 9(b). This is because the sensors in the purple cluster are deployed on the road that goes from south to north, where the sensors in the green cluster are on the opposite direction. Two different directions on the same road may have quite different patterns. These provide strong evidence that the learned stochastic variables well reflect the patterns of time series from different locations.

VI. Conclusion and Outlook

We propose a data-driven, model-agnostic method to turn spatio-temporal agnostic models to spatio-temporal aware models. In particular, we apply the method to enable spatio-temporal aware attention that is capable of modeling complex spatio-temporal dynamics in traffic time series. In addition, we propose an efficient window attention with linear complexity, thus ensuring competitive overall efficiency. Extensive experiments on four datasets show that our proposal outperforms other state-of-the-art methods. A limitation of our proposal is that the learning is based on the assumption that the latent stochastic variables follow Gaussian distributions. In future research, it is of interest to explore methods such as normalizing flows for to employ non-Gaussian stochastic variables.

VII. Acknowledgements

References

[1] R.-G. Cirstea, B. Yang, C. Guo, T. Kieu, and S. Pan, "Towards spatio-temporal aware traffic time series forecasting," in *ICDE*, 2022.

[2] S. B. Yang, C. Guo, and B. Yang, "Context-aware path ranking in road networks," *IEEE Trans. Knowl. Data Eng.*, 2020.

[3] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Distinguishing trajectories from different drivers using incompletely labeled trajectories," in *CIKM*, 2018, pp. 863–872.

[4] S. B. Yang, C. Guo, J. Hu, J. Tang, and B. Yang, "Unsupervised path representation learning with curriculum negative sampling," in *IJCAI*, 2021, pp. 3286–3292.

[5] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *TIST*, vol. 5, no. 3, pp. 1–55, 2014.

[6] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *MDM*, 2018, pp. 125–134.

[7] T. Kieu, B. Yang, C. Guo, R.-G. Cirstea, Y. Zhao, Y. Song, and C. S. Jensen, "Anomaly detection in time series with robust variational quasi-recurrent autoencoders," in *ICDE*, 2022.

[8] D. Campos, T. Kieu, C. Guo, F. Huang, K. Zheng, B. Yang, and C. S. Jensen, "Unsupervised time series outlier detection with diversity-driven convolutional ensembles," *PVLDB*, vol. 15, no. 3, pp. 611–623, 2022.

[9] T. Kieu, B. Yang, C. Guo, C. S. Jensen, Y. Zhao, F. Huang, and K. Zheng, "Robust and explainable autoencoders for time series outlier detection," in *ICDE*, 2022.

[10] S. A. Pedersen, B. Yang, and C. S. Jensen, "Fast stochastic routing under time-varying uncertainty," *VLDB J.*, vol. 29, no. 4, pp. 819–839, 2020.

[11] C. Guo, B. Yang, J. Hu, C. S. Jensen, and L. Chen, "Context-aware, preference-based vehicle routing," *VLDB J.*, vol. 29, no. 5, pp. 1149–1170, 2020.

[12] H. Liu, C. Jin, B. Yang, and A. Zhou, "Finding top-k optimal sequenced routes," in *ICDE*, 2018, pp. 569–580.

[13] S. A. Pedersen, B. Yang, and C. S. Jensen, "Anytime stochastic routing with hybrid learning," *PVLDB*, vol. 13, no. 9, pp. 1555–1567, 2020.

[14] S. B. Yang, C. Guo, J. Hu, B. Yang, J. Tang, and C. S. Jensen, "Temporal path representation learning with weakly-supervised contrastive curriculum learning," in *ICDE*, 2022.

[15] D. Deng, C. Shahabi, U. Demiryurek, L. Zhu, R. Yu, and Y. Liu, "Latent space model for road networks to predict time-varying traffic," in *SIGKDD*, 2016, pp. 1525–1534.

[16] L. Bai, L. Yao, S. S. Kanhere, Z. Yang, J. Chu, and X. Wang, "Passenger demand forecasting with multi-task convolutional recurrent neural networks," in *PAKDD*, 2019, pp. 29–42.

[17] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *ICLR*, 2018, pp. 1–16.

[18] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," in *NeurIPS*, 2020, pp. 1–16.

[19] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *ICONIP*, 2018, pp. 362–373.

[20] J. Hu, B. Yang, C. Guo, C. S. Jensen, and H. Xiong, "Stochastic origin-destination matrix forecasting using dual-stage graph convolutional, recurrent neural networks," in *ICDE*, 2020, pp. 1417–1428.

[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440.

[22] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *IJCAI*, 2019, pp. 1907–1913.

[23] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *IJCAI*, 2018, pp. 3634–3640.

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

[25] C. Park, C. Lee, H. Bahng, Y. Tae, S. Jin, K. Kim, S. Ko, and J. Choo, "ST-GRAT: a novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed," in *CIKM*, 2020, pp. 1215–1224.

[26] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *NeurIPS*, 2019.

[27] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," *WWW*, pp. 499–508, 2018.

[28] M. Li and Z. Zhu, "Spatial-temporal fusion graph neural networks for traffic flow forecasting," in *AAAI*, 2021, pp. 1–7.

[29] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *AAAI*, 2019, pp. 922–929.

[30] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," in *AAAI*, 2020, pp. 914–921.

[31] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[32] R.-G. Cirstea, B. Yang, and C. Guo, "Graph attention recurrent neural networks for correlated time series forecasting." in *MileTS19@KDD*, 2019.

[33] S. Guo, Y. Lin, H. Wan, X. Li, and G. Cong, "Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[34] X. Wu, D. Zhang, C. Guo, C. He, B. Yang, and C. S. Jensen, "AutoCTS: Automated correlated time series forecasting," *PVLDB*, vol. 15, no. 4, pp. 971–983, 2022.

[35] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[36] P. Zhang, X. Dai, J. Yang, B. Xiao, L. Yuan, L. Zhang, and J. Gao, "Multi-scale vision longformer: A new vision transformer for high-resolution image encoding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2998–3008.

[37] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, and H. Xiong, "Spatial-temporal transformer networks for traffic flow forecasting," *arXiv preprint arXiv:2001.02908*, 2020.

[38] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *AAAI*, 2019, pp. 3656–3663.

[39] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *AAAI*, 2017, pp. 1655–1661.

[40] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu, "Traffic flow prediction via spatial temporal graph neural network," in *WWW*, 2020, pp. 1082–1092.

[41] L. Bai, L. Yao, S. Kanhere, X. Wang, Q. Sheng *et al.*, "STG2Seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting," in *AAAI*, 2019, pp. 1981–1987.

[42] J. Chen, X. Qiu, P. Liu, and X. Huang, "Meta multi-task learning for sequence modeling," *AAAI*, 2018.

[43] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, "Urban traffic prediction from spatio-temporal data using deep meta learning," in *SIGKDD*, 2019, pp. 1720–1730.

[44] R.-G. Cirstea, T. Kieu, C. Guo, B. Yang, and S. J. Pan, "EnhanceNet: Plugin neural networks for enhancing correlated time series forecasting," in *ICDE*. IEEE, 2021, pp. 1739–1750.

[45] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic convolution: Attention over convolution kernels," in *CVPR*, 2020, pp. 11 030–11 039.

[46] B. De Brabandere, X. Jia, T. Tuytelaars, and L. Van Gool, "Dynamic filter networks," in *NeurIPS*, 2016, pp. 667—-675.

[47] B. Yang, G. Bender, Q. V. Le, and J. Ngiam, "Condconv: Conditionally parameterized convolutions for efficient inference," in *NeurIPS*, 2019, pp. 1307–1318.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, 2017.

[49] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *ICLR*, 2014.

[50] T. W. Anderson, "An introduction to multivariate statistical analysis," Wiley New York, Tech. Rep., 1962.

[51] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," *arXiv preprint arXiv:1802.04407*, 2018.

[52] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in Statistics*. Springer, 1992, pp. 492–518.

[53] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *JMLR*, vol. 9, no. Nov, pp. 2579–2605, 2008.