# FastSTI: A Fast Conditional Pseudo Numerical Diffusion Model for Spatio-temporal Traffic Data Imputation

Shaokang Cheng, Nada Osman, Shiru Qu, and Lamberto Ballan , *Senior Member, IEEE*

***Abstract*—High-quality spatiotemporal traffic data is crucial for intelligent transportation systems (ITS) and their data-driven applications. Inevitably, the issue of missing data caused by various disturbances threatens the reliability of data acquisition. Recent studies of diffusion probability models have demonstrated the superiority of deep generative models in imputation tasks by precisely capturing the spatio-temporal correlation of traffic data. One drawback of diffusion models is their slow sampling/denoising process. In this work, we aim to accelerate the imputation process while retaining the performance. We propose a fast conditional diffusion model for spatiotemporal traffic data imputation (FastSTI). To speed up the process yet, obtain better performance, we propose the application of a high-order pseudo-numerical solver. Our method further revs the imputation by introducing a predefined alignment strategy of variance schedule during the sampling process. Evaluating FastSTI on two types of real-world traffic datasets (traffic speed and flow) with different missing data scenarios proves its ability to impute higher-quality samples in only six sampling steps, especially under high missing rates (60% $\sim$ 90%). The experimental results illustrate a speed-up of 8.3$\times$ faster than the current state-of-the-art model while achieving better performance.**

***Index Terms*—Traffic data imputation, conditional diffusion model, pseudo numerical methods, fast sampling.**

## I. Introduction

SPATIOTEMPORAL traffic data, acquired by diverse sensing systems, plays a fundamental role in intelligent transportation systems (ITS), since they allow a wide array of applications and decision-making processes [1], [2]. Nevertheless, the common presence of missing data due to equipment failures and transmission errors negatively affects downstream tasks, such as traffic flow prediction. Missing values can lead city planning authorities to rerun experiments to gather necessary data, resulting in additional budgetary expenses and time delays [3]. For decades, missing data imputation techniques have been extensively investigated, mostly trying to identify spatial and temporal correlations from observed data and accurately estimate the missing values [4].

Focusing on imputing traffic data, early traditional methods, exemplified by statistical [5], [6] or classical machine learning (ML) [7], [8] are trivially simple that they neglect the potentially complex interactions between spatial and temporal correlations of traffic conditions. Conversely, deep learning (DL)-based techniques have proven practicality in approximating complex functions and better mining the spatiotemporal evolution patterns. For example, RNNs and their variants [9], [10] are commonly employed to impute missing values. Still, RNN-based approaches present the limitation of assuming sequential relationships within time-series data [11], [12]. Additionally, RNNs cannot exploit parallel processing capabilities and face difficulties directly modeling the interdependence among input data with distinct timestamps.

Among various imputation methods, deep generative models have gained significant popularity. Recently, diffusion models have emerged as the new state-of-the-art method of deep generative models family [13]–[15], surpassing the long-standing dominance of generative adversarial networks (GANs) in diverse of challenging domains [16]. Compared with other probabilistic approaches (e.g., VAEs and GANs), diffusion-based imputation offers stable training and models sophisticated data distributions by sufficient denoising steps [17].

Existing works started to apply Denoising Diffusion Probabilistic Model (DDPM) to impute missing traffic data [18]–[20]. However, these approaches involve iterative procedures with several evaluation steps, which can be time-consuming and inefficient in real-time applications. To guarantee the quality of the generated samples and enhance the model's capacity for representation, a number of denoise iterations are necessary for diffusion-based imputation methods, which lead to higher computational processing. For example, PriSTI [20] experiments on the METR-LA dataset show that diffusion models need around 50 denoising steps for satisfactory estimations, taking $\sim 5s$ to impute average five-minute traffic speed data from 207 sensors. This inevitably limits the real-time performance of missing data imputation.

In this work, we design a fast conditional pseudo-numerical diffusion model for spatiotemporal traffic data imputation, where we apply pseudo-numerical methods and a predefined variance schedule to accelerate the inference time while retaining the imputation precision. Our acceleration technology is general and compatible with most diffusion-based models. To the best of our knowledge, FastSTI is the first to allow real-time application of conditional diffusion models in traffic data imputation with minimal computational cost and negligible

*(Corresponding authors: Shiru Qu)*

Shaokang Cheng and Shiru Qu are with the School of Automation, Northwestern Polytechnical University, Xi'an 710072, China. (e-mail: qushiru@nwpu.edu.cn)

Nada Osman and Lamberto Ballan are with the Department of Mathematics "T. Levi-Civita", University of Padova, Padova 35131, Italy.

loss of accuracy. We can brief our contributions as follows:

1) We put forward and integrate the high-order pseudo-numerical solvers into a conditional diffusion model to improve traffic data imputation, both accuracy and speed.
2) To further accelerate the data imputation process, we tune and utilize a variance schedule that derives a short and effective noise schedule, reducing computation time without significant loss of accuracy.
3) To leverage observational feature knowledge, we use a graph convolution network (GCN) variants to capture the correlation of traffic conditions from both spatial and temporal perspectives.
4) Extensive experiments are conducted to evaluate our proposed imputation method, with different data-missing scenarios and high missing rates, confirming the effectiveness of our FastSTI model.

The remainder of this work is organized as follows: Section II reviews related work. Section III defines the preliminaries. Section IV introduces our proposed method FastSTI. Section V presents experiments and performance discussions, and we conclude in Section VI.

## II. Related Work

### A. Spatiotemporal Traffic Data Imputation

Spatiotemporal traffic data imputation has emerged as a prominent area of study in urban computing. Early conventional methods mainly relied on statistical or classical machine learning (ML) techniques, overlooking the complex spatiotemporal patterns within the city's road network. These methods include but are not limited to Mean, Linear, KNN, MICE, VAR, and KF [5], [7]. Likewise, some studies use matrix and tensor factorization methods, which typically rely on low-order decomposition to estimate or impute missing data [21], [22]. In recent years, DL-based methods have become widely employed when dealing with traffic data imputation tasks that require complex and nonlinear situations. A common approach in DL is to utilize RNNs and their variants, such as LSTMs and GRUs, for time-series imputation [9], [10], [23]. Nevertheless, RNN-based imputation models are time-consuming, sensitive to error propagation during imputation, and struggle to capture dynamic changes effectively.

Recent deep generative-based models have made significant advancements in spatiotemporal traffic data imputation, such as variational autoencoders (VAEs) [24]–[26], generative adversarial networks (GANs) [27], [28], and normalizing flows [29], [30]. As a state-of-the-art class of deep generative models, diffusion models have emerged as strong contenders challenging the long-standing supremacy of GANs [16]. For instance, Tashiro et al. [18] proposed a probabilistic imputation method to directly learn the conditional distribution with conditional score-based diffusion models. Alcaraz et al. [19] put forward a combination of state-space models as effective blocks for capturing long-term dependencies in time series with conditional diffusion models. Also, Liu et al. [20] proposed PriSTI, a global context prior diffusion framework for imputing spatiotemporal data. To enhance accuracy, PriSTI incorporates conditional information, spatiotemporal global correlations, and geographic relationships. However, the above methods pose challenges to the practical application of data imputation due to the extensive inference time. In this work, we propose a fast, high-quality traffic data imputation diffusion model.

### B. Diffusion Models

Diffusion models are a family of probabilistic generative models that proved outstanding performance in various domains, including but not limited to computer vision [31], spatiotemporal data modeling [18], natural language processing [32], and multi-modal learning [33]. Ho et al. [13] propose Denoising Diffusion Probabilistic Model (DDPM), utilizing two Markov chains process of the $T$-steps: 1) a $diffusion\ process$ that perturbs data by adding noise, and 2) a $reverse\ process$ that reconstructs the data from the noise. Given a data distribution $\acute{x}_0 \sim p(\acute{x}_0)$, and $\acute{x}_t$ is the sampled latent variable sequence, with $t = 1, \cdots, T$ denoting the diffusion steps. The $diffusion\ process$ gradually adds standard Gaussian noise into $\acute{x}_0$ until it becomes close to $\acute{x}_t$, while the $reverse\ process$ denoises $\acute{x}_t$ to recover $\acute{x}_0$.

However, the process of generating samples from DDPM requires iterative approaches that involve multiple evaluation steps. Many works have focused on accelerating the sampling process and improving the quality of the resulting samples by considering stochastic differential equations (SDEs) or ordinary differential equations (ODEs). Compared to SDE-slover, ODE-solver brings a more promising approach as the trajectories they solve are deterministic and not influenced by random fluctuations. One example of early work in accelerating diffusion model sampling is Denoising Diffusion Implicit Models (DDIM) [34], considered as a first-order ODE-solver. In extensive experimental investigations, Karras et al. [35] recently demonstrated that high-order numerical solvers achieve a better trade-off between sample quality and speed. By obtaining the ODE form, many numerical solvers can be readily applied to the sampling process. To a certain extent, high-order solvers reduce discretization errors but also increase computational requirements. Instead, our framework enhances the denoising process of the diffusion model by integrating the variants of high-order numerical solvers and presenting an acceleration method that effectively reduces time consumption.

### C. Numerical Methods

As stated above, various numerical methods can provide high-order approaches for solving ODEs [36], such as:

1) Heun's Method: also named improved Euler, it modifies Euler's method from one step into a two-step to improve accuracy.
2) Runge-Kutta Methods (RK): a class of numerical techniques that enhance accuracy by incorporating information from multiple hidden steps.
3) Linear Multi-Step Method (LMS): Similar to but different from RK methods, the linear multi-step method

TABLE I
NOTATION

| Notations | Descriptions |
|---|---|
| $X$ | Road traffic data |
| $\acute{X}$ | Manually selected imputation target |
| $L$ | The length of time steps |
| $N$ | The number of the observation nodes |
| $G$ | Road traffic network |
| $A$ | Adjacency matrix of geographic information |
| $K$ | Graph random walk steps |
| $\chi$ | Conditional observations |
| $T$ | The number of diffusion steps |
| $\alpha_t, \beta_t, \bar{\alpha}_t, \epsilon$ | Constant hyperparameters of diffusion process |
| $\epsilon_\theta$, $\bar{\varphi}_t$, $\bar{\xi}_t$ | Constant hyperparameters of imputation process |
| $T_{acc}$ | The number of accelerated denoising steps |

utilizes previous steps rather than hidden steps to estimate the next step.

However, [37], [38] also showed that these classical numerical methods introduce significant noise at a high speedup ratio, causing the solver to sample data that deviates from the main distribution area. To deal with this issue, Liu et al. [38] designed a pseudo-numerical method for unconditional diffusion models by introducing a nonlinear transfer part as the pseudo-numerical method rather than directly using classical numerical methods. Our approach leverages the pseudo-numerical solvers into a conditional diffusion model to further extract additional prior knowledge, enhancing the imputation accuracy of traffic data.

## III. PRELIMINARIES

Our task is to estimate the missing data or the corresponding distributions in traffic datasets with incomplete observed values. We define the traffic network as a graph $G$, exploit the spatiotemporal correlation within the network, and impute the missing values. Table I summarizes the notations used throughout our work.

Formally, let $X = \{x_1, x_2, \ldots, x_L\}$ be a sequence with shape $R^{L\times N}$, where $L$ represents the length of time steps, and $N$ is the number of observation nodes (e.g., traffic sensors/loop detectors). The observation nodes can be denoted with a directed graph $G = (V, E, A)$, where $V$ and $E$ represent a finite set of nodes and edges, respectively. The adjacency matrix $A \subseteq R^{N\times N}$ is a distance weight matrix, standing for the geographic distance between each pair of observation nodes ($v_i$, $v_j$). Here, we use threshold Gaussian Kernel [39] to obtain the adjacency matrix from the geographic distances among nodes.

To account for missing values, we use a binary mask $M_l \in \{0, 1\}^N$ that indicates which node of $x_l$ are observed in $X$. Specifically, if $m_l^{i,j} = 0$, the corresponding element $x_l^{i,j}$ is missing, while $m_l^{i,j} = 1$ denotes that $x_l^{i,j}$ is an observed value. To handle missing data in practical scenarios where ground truth is unavailable, we manually select imputation targets $\acute{X}$ from the observed data and then use the binary mask $M_l \in \acute{\{0, 1\}}^N$ to identify these targets. In this way, we can assess the efficiency of our imputation technique under real-life conditions.

## IV. METHODOLOGY

In this section, we describe the basic ideas of our proposed Fast Conditional Pseudo Numerical Diffusion Model for Spatio-temporal Traffic Data Imputation (FastSTI). Figure 1 illustrates the architecture of our proposed method. We build our methodology on top of the state-of-the-art PriSTI model [20]. Different from PriSTI [20], we propose a high-order conditional pseudo-numerical method to improve the sampling quality in the reverse process of the diffusion model. Meanwhile, we explore an accelerated method by using variance scheduling to address limitations in the inference time of diffusion model-based baselines. In addition, a variant of GCN (Diff-GCN) is adopted to enhance the extraction of local spatial correlations of traffic patterns (i.e., traffic speed and flow), in replacement of the direct message passing approach used in PriSTI.

### A. Masking Strategy of Imputation Targets

Given an observed sequence $X$, we split it into two parts: one represents the imputation target $\acute{X}$, while the other represents the observed values serving as conditional observations. To simulate different real-life scenarios of missing traffic data, following [10], we consider two masking strategies:

- **Block-missing scenario:** Missing values occur in contiguous blocks over time. We randomly mask $5\%$ of the available data and adopt simulated failures with a probability of $0.15\%$ for each node/sensor. The duration of each failure is sampled uniformly from the interval $[min\_steps, max\_steps]$, where $min\_steps$ and $max\_steps$ correspond to the length of time steps.
- **Point-missing scenario:** Random occurrence of missing values, where $25\%$ of observations are masked in a random manner.

### B. Conditional Diffusion Model

Our conditional diffusion model utilizes the observed values and geographic location data of all sensors as inputs. This input information is processed through a coarse interpolation and conditional feature prior module to model and extract the spatiotemporal correlations of traffic patterns (i.e., traffic speed and flow). Subsequently, the noise prediction module performs reverse denoising using the aforementioned conditional feature information to generate the imputation output.

**Coarse Interpolation.** To provide rough but efficient interpolated conditional information denoted as $\chi$, Linear interpolation (Lin-TIP) is employed to fill in the missing data at each node. This approach relies on the uniform distribution of missing data to ignore the randomness of time series but retains certain spatiotemporal relations. We can also observe the baseline result of Lin-TIP from Table V. Meanwhile, Lin-TIP, a simple architecture, meets the requirements for real-time performance.

**GCN-based Conditional Feature Prior.** The linear interpolation method assumes uniform and linear changes in traffic states. However, traffic data exhibits dynamic temporal dependencies, and the flow of different regions/interactions

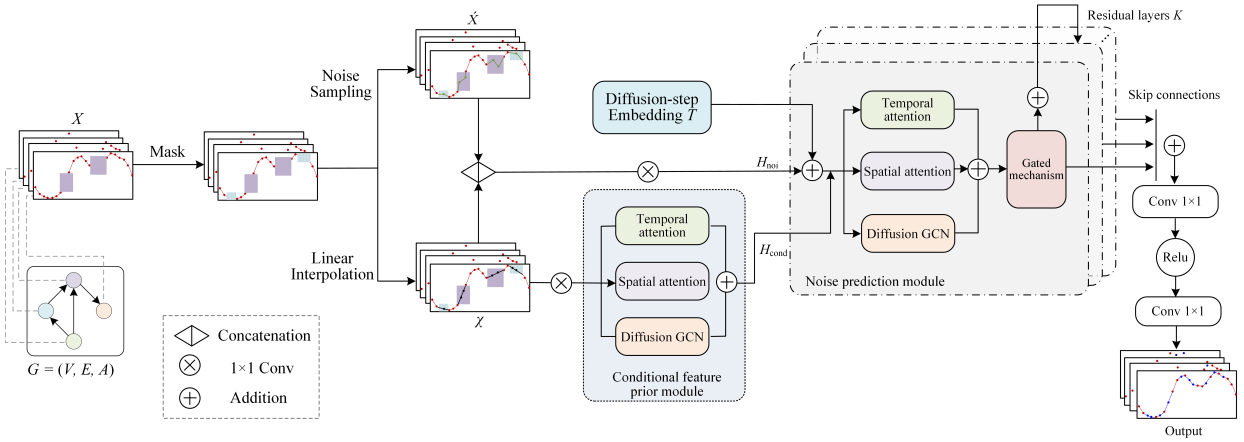



Fig. 1. Proposed FastSTI model architecture. In FastSTI, we take observed values and geographic location information as input. Our approach uses linear interpolation and leverages the conditional feature prior module to model the prior spatiotemporal context. Afterwards, the prior feature weights are obtained and fed into the noise prediction module to help predict noise.

affects each other, making linear interpolation inadequate for capturing the nonlinear and random patterns in real-life traffic conditions [40]. Therefore, it is a necessity to utilize a learnable module that can adapt to such non-linearity and randomness in real-life traffic patterns. To address the problem, we adopt a *conditional feature prior module* $\rho(\cdot)$, which takes the interpolated information $\chi$ and the adjacency matrix $\mathcal{A}$ to model the nonlinear conditional information, extracting global spatial and temporal correlations as well as local geographic correlations of road traffic patterns, as illustrated in Eq. 1, where the input $\mathcal{H}$ denotes a $1d$ convolution of the interpolated data ($\mathcal{H} = Conv(\chi)$). Similar to [20], the global temporal correlation is captured using a transformer-based self-attention module $\phi_{Tem}(\mathcal{H})$ (Eq. 2), while another global self-attention module $\phi_{Spa}(\mathcal{H})$ extracts the global spatial correlations, as in Eq. 3. To better model the local geographic correlations, our conditional prior module replaces the message-passing network in [20] with a graph convolution module $\phi_{\mathrm{DGCN}}(\mathcal{H}, \mathcal{A})$), as in Eq. 4.

$$\rho(\mathcal{H}, \mathcal{A}) = \mathrm{MLP}(\phi_{Tem}(\mathcal{H}) + \phi_{Spa}(\mathcal{H}) + \phi_{\mathrm{DGCN}}(\mathcal{H}, \mathcal{A})) \tag{1}$$

$$\phi_{Tem}(\mathcal{H}) = \mathrm{Norm}(Attn_{Tem}(\mathcal{H}) + \mathcal{H}) \tag{2}$$

$$\phi_{Spa}(\mathcal{H}) = \mathrm{Norm}(Attn_{Spa}(\mathcal{H}) + \mathcal{H}) \tag{3}$$

$$\phi_{\mathrm{DGCN}}(\mathcal{H}, \mathcal{A}) = \mathrm{Norm}(\mathrm{DiffGCN}(\mathcal{H}, \mathcal{A}) + \mathcal{H}) \tag{4}$$

In contrast to the message-passing network (MP) employed in PriSTI [20], our framework captures the local geographic correlations with an improved graph convolution network module named Diffusion-GCN (Diff-GCN). The local geospatial dependency in traffic flow is modeled by correlating it with a diffusion flow process, described by [41], which effectively extracts the random nature of traffic dynamics, in addition to featuring a more lightweight structure during the diffusion process, meeting real-time requirements. Additionally, Diff-GCN benefits from a bi-directional random walk $K$ strategy, providing enhanced flexibility in capturing influences from upstream and downstream traffic conditions (e.g., speed or flow).

Specifically, the key operation of diffusion graph convolution over the coarse interpolated data $\chi$ is defined in Eq. 5, where $A$ denotes an adjacency matrix (road distance-based node matrix), $K$ is the step of graph random walk, $\varrho \in [0, 1]$ is the graph coefficient, $\theta_k$ is the parameter of the convolutional filter, and $D_g$ and $D_c$ are the transition matrices of the graph diffusion process and the converse one. Here, $D = diag(A*1)$, matrix $1 \in R^N$ represents all element is one.

$$\mathrm{DiffGCN}(\mathcal{H}, \mathcal{A}) = \sum_{k=0}^{K} \left( \varrho_{k \geq 1} \left( \theta_k^1 (D_g A)^k + \theta_k^2 (D_r A)^k \right) \right) \chi \tag{5}$$

Simply put, Diff-GCN generates a group of node features as output after aggregating graph information from the self-node and its $K$-hop neighbors. Notably, the number of neighbors in the graph random walk $K$ is a hyperparameter to be tuned, where a large $K$ value can enable graph convolution to capture more geospatial information. The learning process also becomes more complex and the computation time increases when larger convolution filters are chosen.

Once the conditional feature prior module $H_{\mathrm{cond}}$ is established, these coarse yet useful conditional features are fed into the noise prediction module to facilitate the learning of spatiotemporal correlations of road traffic patterns.

**Noise Prediction Module.** The noise prediction module is designed to utilize conditional information to predict the missing values, as shown in Fig. 1. The module takes two inputs: 1) The conditional prior $H_{cond}$; 2) Noise information $H_{noi} = Conv(\chi || \hat{X})$, where $Conv(.)$ is the $1 \times 1$ convolution, $(||)$ represents concatenation, and $\hat{X}$ is the data sequence with the missing data sampled from a standard Gaussian noise. The temporal, spatial, and geographic modules of the noise prediction are the same modules of the conditional feature prior. However, the noise information $H_{noi}$ would not provide an accurate representation of real-life traffic data; therefore, the conditional information $H_{cond}$ is used for both the query

**Algorithm 1** Training procedure

**Require:** incomplete observed data $X$, the noise levels $\bar{\alpha}_t$, the adjacency matrix $A$, the number of iteration $N_i^t$, the diffusion steps $T$.
**for** $i = 1$ to $N_i^t$ **do**
  Sample $\mathrm{x_T} \sim \mathrm{q_{data}}, \boldsymbol{\epsilon} \sim (0, \boldsymbol{I})$,
  $t \sim \mathrm{Uniform}\left(\{0, \ldots, T-1\}\right)$
  Compute $\acute{\mathrm{X}}_\mathrm{t} = \sqrt{\bar{\alpha}_\mathrm{t}}\mathrm{x}_0 + \sqrt{1-\bar{\alpha}_\mathrm{t}}\epsilon$
  $\mathcal{L}_t = || \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\acute{X}_t, \chi, A, t) ||^2$
  Take gradient descent step on $\nabla_\theta \mathcal{L}_\mathrm{t}$
**end for**

**Algorithm 2** Imputation (sampling) process of FastSTI-4

**Require:** observed data $X$, adjacency matrix $A$, #iterations $N_i^t$, noise predictor $\epsilon_\theta$, diffusion steps $T$.
$x_T \sim \mathcal{N}(0, \boldsymbol{I})$
**for** $t = T-1, T-2, T-3$ **do**
  $x_t, e_t = \mathrm{PRK4}(x_{t+1}, \chi, A, t+1, t)$
**end for**
**for** $t = T-4, \ldots, 0$ **do**
  $x_t, e_t = \mathrm{PLMS4}(x_{t+1}, \{e_p\}_{p>t}, \chi, A, t+1, t)$
**end for**
**return** $x_0$

and value of the temporal attention $\phi_{Tem}$ and spatial attention $\phi_{Spa}$, while $H_{noi}$ is used for the key.

The output of each layer in the noise prediction module is split into a residual connection and skip connections. The residual connection is the input of the next layer, and the skip connections of each layer are added and fed into a two-layers $1d$ convolution to obtain the output of the noise prediction module, where the output contains only the value of the imputation target.

### C. Training Procedure

Given an observed sequence $X$ and split into two parts: one part represents the imputation target $\acute{X}$ which is generated by the masking strategies (i.e., block-missing and point-missing strategy), while the remaining observed values serve as conditional observations $\chi$.

Next, given the imputation target $\acute{X}$ and the interpolated conditional observations $\chi$, we sample the imputation target $\acute{X}$ and train the noise prediction model $\epsilon_\theta$ by minimizing the loss function. Where the noise predictor minimizes the loss function $\mathcal{L}_t$, defined in Eq. 6, where the imputation target $\acute{X}_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$; $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, $\alpha_t = 1 - \beta_t$. The training procedure is summarized in Algorithm 1.

$$\min \mathcal{L}_t = \min \mathbb{E}_{\acute{X} \sim q(\acute{X}_0), \epsilon \sim N(0,I)} \, || \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\acute{X}_t, \chi, A, t) ||^2 \tag{6}$$

### D. Conditional Pseudo-Numerical Methods

As described earlier, we aim to speed up the imputation process and improve its quality. Inspired by [38], we introduce the pseudo-numerical solvers to our conditional diffusion model during the reverse/sampling process. Different from the baseline models PriSTI [20] and CSDI [18] that directly use reverse sampling calculation of DDPM, our approach leverages the higher-order numerical solver to sample the distribution of data and avoid the influence of random noise caused by DDPM, thereby enhancing the quality of data generation.

In this work, we apply two kinds of pseudo-numerical methods for the conditional diffusion model in our imputation task: **FastSTI-2** ($2^{nd}$-order) and **FastSTI-4** ($4^{th}$-order). As DDPM can be considered a special case of DDIM [34], we provide the following steps to develop the high-order pseudo-numerical sampling method in the conditional diffusion model. First, the reverse process of DDIM is transformed into an ordinary differential equation (ODE) form. Then, the nonlinear transfer part is combined with three classical numerical techniques, namely Heun's methods, Runge-Kutta methods (RK), and Linear Multi-Step methods (LMS). Meanwhile, the conditional information is built into the pseudo-numerical diffusion model for our task of traffic data imputation.

Mathematically, the reverse process of DDIM is defined in Eq. (7). To construct the ODE form of the diffusion model, the equation is reformulated by subtracting $x_t$ from both sides. This reformulation makes it equivalent to a numerical step in solving the ODE, as demonstrated in Eq. (8).

$$x_{t-1} = \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}} \left(x - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, t)\right) + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta(x_t, t) \tag{7}$$

$$x_{t-\Delta t} - x_t = (\bar{\alpha}_{t-\Delta t} - \bar{\alpha}_t) \left( \frac{x_t}{\sqrt{\bar{\alpha}_t}\left(\sqrt{\bar{\alpha}_{t-\Delta t}} + \sqrt{\bar{\alpha}_t}\right)} - \frac{\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}\left(\sqrt{(1-\bar{\alpha}_{t-\Delta t})\,\bar{\alpha}_t} + \sqrt{(1-\bar{\alpha}_t)\,\bar{\alpha}_{t-\Delta t}}\right)} \right) \tag{8}$$

Then, the discrete-time step $t-1$ in Eq. (7) is replaced with a continuous version represented by $t - \Delta t$, and by letting $\Delta t$ tends to 0, the ODE becomes:

$$\frac{dx}{dt} = -\bar{\alpha}'(t) \left( \frac{x(t)}{2\bar{\alpha}(t)} - \frac{\epsilon_\theta(x(t), t)}{2\bar{\alpha}(t)\sqrt{1-\bar{\alpha}(t)}} \right) \tag{9}$$

After obtaining the ODE, the pseudo-numerical method categorizes the classical numerical methods (Heun's, RK, and LMS) into two components:

1) *The gradient part*, responsible for determining the gradient at each step, e.g., $4^{th}$-order linear multi-steps gradient part $f' = \frac{\Delta t}{24}(55f_t - 59f_{t-\Delta t} + 37f_{t-2\Delta t} - 9f_{t-3\Delta t})$.
2) *The transfer part*, as shown in Eq. (10), which generates the result for the next step, i.e., $x_{t-\Delta t} = x_t + \Delta t f'$.

All numerical methods share the same transfer part, while their gradient parts differ. The transfer part $x_{t-\Delta t}$ is obtained by $\nu$, rewriting Eq. (8) into Eq. (10), where $\chi$ represents the conditional observations. While the gradient part is defined for each numerical method in Table II.

TABLE II
GRADIENT EQUATIONS OF THE DIFFERENT PSEUDO NUMERICAL METHODS

$2^{nd}$-order pseudo linear multi-step (PLMS2)

$$\begin{cases} e_t = \epsilon_\theta\left(\acute{x}_t, \chi, A, t\right), \\ e'_t = \frac{1}{2}(3e_t - e_{t-\Delta t}), \\ x_{t-\Delta t} = \nu(\acute{x}_t, \chi, \acute{e_t}, A, t, t-\Delta t). \end{cases}$$

$2^{nd}$-order pseudo Heun's (PH2)

$$\begin{cases} e_t^1 = \epsilon_\theta\left(\acute{x}_t, \chi, A, t\right), \\ x_t^1 = \nu(\acute{x}_t, \chi, e_t^1, A, t, t-\Delta t), \\ e_t^2 = \epsilon_\theta\left(\acute{x}_t^1, \chi, A, t-\Delta t\right), \\ e'_t = \frac{1}{2}(e_t^1 + e_t^2), \\ x_{t-\Delta t} = \nu(\acute{x}_t, \chi, \acute{e_t}, A, t, t-\Delta t). \end{cases}$$

$4^{th}$-order pseudo linear multi-step (PLMS4)

$$\begin{cases} e_t = \epsilon_\theta\left(\acute{x}_t, \chi, A, t\right), \\ e'_t = \frac{1}{24}(55e_t - 59e_{t-\Delta t} + 37e_{t-2\Delta t} - 9e_{t-3\Delta t}), \\ x_{t-\Delta t} = \nu(\acute{x}_t, \chi, \acute{e_t}, A, t, t-\Delta t). \end{cases}$$

$4^{th}$-order pseudo Runge-Kutta (PRK4)

$$\begin{cases} e_t^1 = \epsilon_\theta\left(\acute{x}_t, \chi, A, t\right), \\ x_t^1 = \nu(\acute{x}_t, \chi, e_t^1, A, t, t-\frac{\Delta t}{2}), \\ e_t^2 = \epsilon_\theta(\acute{x}_t^1, \chi, A, t-\frac{\Delta t}{2}), \\ x_t^2 = \nu(\acute{x}_t, \chi, e_t^2, A, t, t-\frac{\Delta t}{2}), \\ e_t^3 = \epsilon_\theta(\acute{x}_t^2, \chi, A, t-\frac{\Delta t}{2}), \\ x_t^3 = \nu(\acute{x}_t, \chi, e_t^3, A, t, t-\Delta t), \\ e_t^4 = \epsilon_\theta\left(\acute{x}_t^3, \chi, A, t-\Delta t\right), \\ e'_t = \frac{1}{6}(e_t^1 + 2e_t^2 + 2e_t^3 + e_t^4), \\ x_{t-\Delta t} = \nu(\acute{x}_t, \chi, \acute{e_t}, A, t, t-\Delta t). \end{cases}$$

$$\nu\left(x_t, \epsilon_t, \chi, A, t, t-\Delta t\right) = \frac{\sqrt{\bar{\alpha}_{t-\Delta t}}}{\sqrt{\bar{\alpha}_t}} x_t - \frac{\left(\bar{\alpha}_{t-\Delta t} - \bar{\alpha}_t\right)}{\sqrt{\bar{\alpha}_t}\left(\sqrt{\left(1-\bar{\alpha}_{t-\Delta t}\right)\bar{\alpha}_t} + \sqrt{\left(1-\bar{\alpha}_t\right)\bar{\alpha}_{t-\Delta t}}\right)}\epsilon_t \tag{10}$$

Algorithm 2 shows an example of our FastSTI-4, where we initially adopt the 4th-order pseudo Runge-Kutta (PRK4) method to obtain the results of the first three steps, followed by the utilization of the $4^{th}$-order pseudo-linear multi-step method (PLMS4) to compute the remaining. Similarly, FastSTI-2 employs the 2nd-order pseudo Heun's (PH2) to obtain the results of the first two steps, followed by utilizing the $2^{th}$-order pseudo-linear multi-step method (PLMS2) to calculate the remaining. Our conditional pseudo-numerical methods are compatible with many diffusion-based models (e.g., PriSTI [20]). We leverage one of the sampling methods to the baseline model in the experiments (see Sec. V-E).

### *E. Accelerated Imputation.*

The application of the reverse process of DDPM for denoising suffers from the drawback of being time-consuming due to the necessity for numerous denoising steps to ensure the quality of the generated samples, see the classical denoising steps (gray part) of Fig. 2. Some typical diffusion-based baselines, such as PriSTI [20] and CSDI [18], with 50 reverse steps, and SSSD [19], with 200 steps, use DDPM's sampling way. Therefore, these baseline models cannot meet real-time requirements.

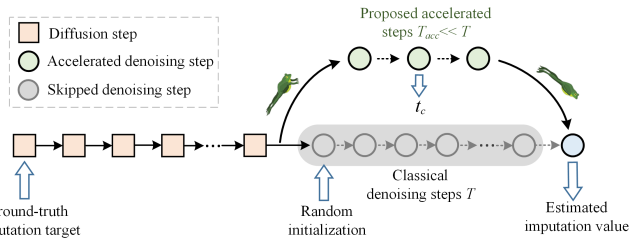



Fig. 2. Accelerated imputation. FastSTI utilizes "schedule alignment" to estimate the denoised distribution, replacing multiple classical denoising steps, thereby accelerating inference without significant loss of accuracy.

**Algorithm 3** Accelerated imputation of FastSTI-4

Sample $x_{T_{acc}} \sim \mathcal{N}(0, \boldsymbol{I})$
**for** $c = T_{acc}-1, T_{acc}-2, T_{acc}-3$ **do**
  $x_c, e_c = \text{PRK4}(x_{c+1}, \chi, A, c+1, c)$
**end for**
**for** $c = T_{acc}-4, \ldots, 0$ **do**
  $x_c, e_c = \text{PLMS4}(x_{c+1}, \{e_p\}_{p>c}, \chi, A, c+1, c)$
**end for**
**return** $x_0$

To accelerate the imputation (sampling) process, inspired by [42], we introduce a "schedule alignment" approach that utilizes a predefined number of $T_{acc}$-steps to minimize the imputation time without significant loss of quality. As shown in Fig. 2, the key concept is to align the original $T$-steps reverse process into a condensed $T_{acc}$-steps process using a predefined variance schedule. In this way, our method is able to skip several reverse steps to reduce denoising iterations. Algorithm 3 summarizes the proposed accelerated sampling (imputation) procedure.

Formally, given the steps of $T_{acc} \ll T$ in the imputation process and a predefined variance schedule $\{\xi_t\}_{t=1}^{T_{acc}}$. Different from the training variance schedule $\{\beta_t\}_{t=1}^{T}$, the $\{\xi_t\}_{t=1}^{T_{acc}}$, with a manual setting of the values, we can calculate the corresponding constants as follows:

$$\varphi_t = 1 - \xi_t,\ \bar{\varphi}_t = \prod_{s=1}^{t} \varphi_c,\ \widetilde{\xi}_t = \frac{1-\bar{\varphi}_{t-1}}{1-\bar{\varphi}_t}\xi_t \tag{11}$$

Our objective is to determine and interpolate the value of $\sqrt{\bar{\varphi}_c}$ between the training noise levels: $\sqrt{\bar{\alpha}_t}$ and $\sqrt{\bar{\alpha}_{t+1}}$, such that $\sqrt{\bar{\varphi}_c}$ closely approximates $\sqrt{\bar{\alpha}_t}$. Afterward, we obtain the aligned diffusion step $t$ by calculating the floating-point $t_c$, as in Eq. (12).

$$t_c = t + \frac{\sqrt{\bar{\alpha}_t} - \sqrt{\bar{\varphi}_c}}{\sqrt{\bar{\alpha}_t} - \sqrt{\bar{\alpha}_{t+1}}} \tag{12}$$

It is worth highlighting that our acceleration approach is general and compatible with most diffusion-based models,

TABLE III
STATISTICS OF DATASETS

| | Traffic Speed | | Traffic Flow | |
|---|---|---|---|---|
| Datasets | METR-LA | PEMS-BAY | PeMS04 | PeMS08 |
| *Nodes (Sensors)* | 207 | 325 | 307 | 170 |
| *Edges* | 1515 | 2694 | 340 | 275 |
| *Total Timestamps* | 34272 | 52116 | 16992 | 17856 |
| *Time Span* | 01/03/2012 - 27/06/2012 | 01/01/2017 - 30/06/2017 | 01/01/2018 - 28/02/2018 | 01/07/2016 - 31/08/2016 |
| *Time Interval* | 5 minutes | | | |
| *Daily Range* | 00:00 - 24:00 | | | |

including PriSTI [20]. To prove this point, we also apply the proposed acceleration technique to the baseline model in the experiments (see Section V-F).

## V. EXPERIMENTS

### A. Datasets

We conduct our experiments on two types of traffic imputation tasks: traffic speed and traffic flow imputation. Table III describes the statistics of these publicly available real-life datasets. Further details about datasets are listed as follows:

1) *Traffic Speed Data*: METR-LA and PEMS-BAY [41]. The traffic speed data is aggregated and reported by the PeMS system at every 5-minute time intervals. METR-LA comprises four months of traffic speed statistics collected from 207 loop detectors installed in the highway of Los Angeles County. Similarly, PEMS-BAY holds six months of traffic speed data from 325 sensors in the Bay Area.

2) *Traffic Flow Data*: PEMS04 and PEMS08 [43]. PEMS04 includes two months of traffic flow data collected from 307 detectors installed in the San Francisco Bay Area. PEMS08 contains two months of traffic flow data from 170 sensors in San Bernardino. Both datasets are aggregated every 5 minutes.

### B. Evaluation Metrics

The data sets are split into $70\%$ for training, $10\%$ for validation, and $20\%$ for testing. The evaluation metrics adopted are Mean Absolute Error (MAE), as in Eq. (13); Mean Square Error (MSE)/Root Mean Square Error(RMSE), described in Eq. (14) and Eq. (15), respectively; and Continuous Ranked Probability Score (CRPS) [44]. MAE and MSE/RMSE assess the errors between targets and imputed values ($e = x_i - \acute{x}_i$), with conditional masking ($m_{eval}$) applied to $x_i$ and $\acute{x}_i$.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|(x_i - \acute{x}_i) \odot m_{eval}| \tag{13}$$

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}((x_i - \acute{x}_i) \odot m_{eval})^2 \tag{14}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}((x_i - \acute{x}_i) \odot m_{eval})^2} \tag{15}$$

CRPS reflects the compatibility of an estimated probability distribution $P$ with the observed values $x$. CRPS Eq. (16) is defined as the integral of the quantile loss $\Lambda_\omega(P^{-1}(\omega), x) = (\omega - \mathbb{1}_{x<P^{-1}(\omega)})(x - P^{-1}(\omega))$, where $P^{-1}(\omega)$ is the $\omega-$quantile of distribution $P$, $\omega \in [0, 1]$ represents the quantile levels, and $\mathbb{1}$ is the indicator function. Following the same setting in [18], 100 samples are generated to approximate the distribution of missing values. We compute quantile losses for discretized quantile levels with 0.05 ticks, defined in Eq. (17), defining $\mathrm{CRPS}(P, \widetilde{X})$ as the average CRPS at each imputed point as in Eq. (18).

$$\mathrm{CRPS}(P^{-1}, x) = \int_0^1 2\Lambda_\omega(P^{-1}(\omega), x)d\omega \tag{16}$$

$$\mathrm{CRPS}(P^{-1}, x) \simeq \sum_{i=1}^{19} 2\Lambda_{i*0.05}(P^{-1}(i*0.05), x)/19 \tag{17}$$

$$\mathrm{CRPS}(P, \widetilde{X}) = \frac{\sum_{\widetilde{x}\in\widetilde{X}} \mathrm{CRPS}(P^{-1}, \widetilde{x})}{\left|\widetilde{X}\right|} \tag{18}$$

### C. Implementation Details

As explained in our methodology, we apply two missing data scenarios: 1) Block-missing, masking $5\%$ of data in the range of $[12, 48]$ time steps, with a node-failure probability of $15\%$; 2) Point-missing, masking randomly $25\%$ of the points. Noting that all datasets include initially missing values (i.e., 8.10% in METR-LA, 0.02% in PEMS-BAY, 1.59% in PEMS04, and 0.35% in PEMS08), except for with manually simulated anomalies. Performance evaluation is conducted primarily on the manually masked test set.

TABLE IV
FASTSTI HYPERPARAMETERS

| Hyperparameter | Value |
|---|---|
| Epochs | 200 |
| Batch size | 16 |
| Sequence length $L$ | 24 |
| Learning rate | $1 \times 10^{-3}$ |
| Weight decay | $1 \times 10^{-6}$ |
| Residual layers | 4 |
| Residual channels $r$ | 64 |
| Self-attention heads $h$ | 8 |
| Temporal embedding dim $m$ | 128 |
| Graph random walk step $K$ | 2 |
| Graph coefficient $\varrho$ | 0.1 |
| Diffusion Schedule | Quadratic |
| The minimum noise level $\beta_1$ | 0.0001 |
| The maximum noise level $\beta_T$ | 0.2 |
| Diffusion steps $T$ | 50 |
| Accelerated denoising steps $T_{acc}$ | 6 |
| Variance Schedule | $\{0.0001, 0.001, 0.2, 0.3, 0.5, 0.9\}$ |

Table IV summarizes our hyperparameters, where the model is trained for 200 epochs with a batch size of 16, learning rate of $10^{-3}$, and Adam optimizer. The diffusion model noise schedule is Quadratic, and we utilize user-defined variance schedules $\{0.0001, 0.001, 0.2, 0.3, 0.5, 0.9\}$.

All the experiments are conducted on Intel(R) Xeon(R) W-2133 CPU @3.60GHz and NVIDIA GeForce 3080Ti GPU

12GB. We implement our FastSTI model in Python 3.7 using Pytorch 1.13.0.

### D. Baselines

We compare our FastSTI model with seventeen baseline models, including statistical (Mean, KNN, Linear InTerPolation (Lin-LTP)), classical ML (MICE, Vector AutoRegression (VAR), Kalman Filter (KF)), low-matrix factorization (TRMF, BATF), deep autoregressive (BRITS, GRIN), and deep generative models (V-RIN, GP-VAE, rGAIN, CSDI, SSSD, PriSTI) in the missing data imputation domain. We provide a concise overview of the baseline models as follows:

(1) Mean: using node-level average to impute missing values. (2) KNN [5]: estimating missing values with observation values from the nearest neighbors based on a similarity measure. (3) Lin-LTP: performing linear interpolation on the time-series data for each node. (4) KF: using Kalman filtering for imputing missing values in temporal observations. (5) MICE [7]: a multiple imputation model with chained equations. (6) VAR: a vector autoregressive single-step-ahead predictor. (7) TRMF [21]: a temporal regularized matrix factorization framework for imputation. (8) BATF [22]: a Bayesian augmented tensor factorization model for imputing missing traffic data. (9) BRITS [9]: a method that uses a bidirectional RNN to impute missing values. (10) GRIN [10]: a framework for multivariate time series imputation by bidirectional graph recurrent neural network. (11) V-RIN [45]: a variational autoencoder framework with a recurrent neural network and a modified loss function to impute the uncertainty of missing values. (12) GP-VAE [24]: an architecture that combines the variational autoencoder and the Gaussian process to impute missing values. (13) rGAIN [27]: a GAN-based method that employs a bidirectional recurrent encoder and decoder architecture. (14) CSDI [18]: a model to impute multivariate time series with conditional score-based diffusion models, leveraging Transformer to extract the features. (15) SSSD [19]: a diffusion-based method using structured state space models for time-series imputation. (16) PriSTI [20]: a conditional diffusion framework for spatio-temporal data imputation with enhanced feature knowledge modeling. (17) PriSTI variant: PriSTI adopts *our $4^{th}$-order conditional pseudo-numerical sampling method* instead of their first-order one from DDPM.

### E. Imputation Performance

First, we compare our model to the seventeen baselines. Table V reports the MAE and MSE/RMSE comparison on two traffic speed datasets (METR-LA and PEMS-BAY) and two traffic flow datasets (PEMS04 and PEMS08), while Table VI reports the CRPS metric. FastSTI, using only 6 reverse steps, outperforms all of the baselines, producing more realistic imputation. Focusing on the comparison between FastSTI and PriSTI [20], we have a better performance, proving the effectiveness of the proposed high-order conditional pseudo-numerical method in generating higher-quality samples. Notably, traffic flow is more challenging than traffic speed because its fluctuations are more significant and random over time. Our approach enhances overall performance on traffic flow datasets (PEMS04 and PEMS08) by utilizing higher-order pseudo-numerical methods that better handle noise and uncertainty in the data. Conversely, PriSTI [20], CSDI [18], and SSSD [19] exhibit limited effectiveness, attributed to their first-order numerical sampling from DDPM [13], which struggles with the complexities of traffic patterns. FastSTI also benefits from the Diff-GCN features extractor, in contrast to the message-passing neural network (MPNN) used in the best baseline PriSTI. Meanwhile, we applied the proposed sampling method on PriSTI, called the PriSTI variant, and the results show that it can help improve the interpolation performance. Additionally, the comparison between FastSTI-2 and FastSTI-4 favors FastSTI-4, which makes sense, as the imputation quality can improve as the order of the numerical method increases.

To comprehensively evaluate the imputation performance under different missing rates, we report missing rates from 10% to 90% on the METR-LA and PEMS04 datasets in Figures 3 and 4. Figure 3a and 4a show such an effect with the block-missing strategy, and Figure 3b and 4b show the impact of increasing the point-missing rate. Intuitively, as the rate of missing values increases, data imputation becomes more challenging due to the reduced availability of observed values and the increased complexity of extracting spatiotemporal correlation of traffic conditions. We can see that the proposed FastSTI-4 (6) consistently outperforms baseline models in terms of imputation performance, regardless of the missing rate. 1) As show in Figures 3a and 3b, FastSTI outperforms the competitors by up to 47.11% in MAE (FastSTI-4 vs. BRITS block-missing at 90% rate) and 42.4% in MAE (FastSTI-4 vs. BRITS point-missing at 90% rate). Compared with the best baseline model PriSTI, FastSTI-4 reduces the MAE by up to 17.96% (block-missing at 90% rate) and 7.3% (point-missing at 90% rate). 2) From a block-missing rate of 10% to 90% (see Fig. 4a), the error of the FastSTI increased by 18.01%, while that of the PriSTI increased by 19.47%. FastSTI is less influenced by changes in the missing rate and has demonstrated better robustness than PriSTI. These results also prove the effectiveness of FastSTI in keeping better accurate imputation, especially under high missing rates (60% $\sim$ 90%), by effectively capturing the spatial and temporal correlations among traffic speed observations across various locations and periods. Moreover, compared to baseline models, our advanced conditional pseudo-numerical method can extract additional features from observed values as the missing rate increases.

### F. Time Performance

We evaluate the inference time of our proposed FastSTI from two perspectives: 1) with and without our acceleration technique, and 2) with two different random walk steps $K \in \{2, 6\}$. The detailed settings are as follows.

1) *Without acceleration*:

- **CSDI [18]**. CSDI with 50 reverse steps, following the original paper settings.
- **PriSTI [20]**. PriSTI with 50 reverse steps, following the original paper settings.
- **FastSTI-2 ($K$=6) w/o acc**. FastSTI-2 with 50 reverse steps and $K$=6 of Diff-GCN.

TABLE V
MAE AND MSE/RMSE COMPARISON WITH THE BASELINES [**BOLD** = BEST, AND UNDERLINE = SECOND BEST]. (6) REPRESENTS THAT OUR ACCELERATED METHODS ONLY REQUIRES 6 REVERSE STEPS DURING THE INFERENCE.

| Method | METR-LA | | | | PEMS-BAY | | | |
|---|---|---|---|---|---|---|---|---|
| | Block-missing (16.52%) | | Point-missing (31.09%) | | Block-missing (9.20%) | | Point-missing (25.01%) | |
| | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE |
| Mean | 7.48±0.00 | 139.54±0.00 | 7.56±0.00 | 142.22±0.00 | 5.46±0.00 | 87.56±0.00 | 5.42±0.00 | 86.59±0.00 |
| KNN | 7.79±0.00 | 124.61±0.00 | 7.88±0.00 | 129.29±0.00 | 4.30±0.00 | 49.90±0.00 | 4.30±0.00 | 49.80±0.00 |
| Lin-ITP | 3.26±0.00 | 33.76±0.00 | 2.43±0.00 | 14.75±0.00 | 1.54±0.00 | 14.14±0.00 | 0.76±0.00 | 1.74±0.00 |
| KF | 16.75±0.00 | 534.69±0.00 | 16.66±0.00 | 529.96±0.00 | 5.64±0.00 | 93.19±0.00 | 5.68±0.00 | 93.32±0.00 |
| MICE | 4.22±0.05 | 51.07±1.25 | 4.42±0.07 | 55.07±1.46 | 2.94±0.02 | 28.28±0.37 | 3.09±0.02 | 31.43±0.41 |
| VAR | 3.11±0.08 | 28.00±0.76 | 2.69±0.00 | 21.10±0.02 | 2.09±0.10 | 16.06±0.73 | 1.30±0.00 | 6.52±0.01 |
| TRMF | 2.96±0.00 | 22.65±0.13 | 2.86±0.00 | 20.39±0.02 | 1.95±0.01 | 11.21±0.06 | 1.85±0.00 | 10.03±0.00 |
| BATF | 3.56±0.01 | 35.39±0.03 | 3.58±0.01 | 36.05±0.02 | 2.05±0.00 | 14.48±0.01 | 2.05±0.00 | 14.90±0.06 |
| BRITS | 2.34±0.01 | 17.00±0.14 | 2.34±0.00 | 16.46±0.05 | 1.70±0.01 | 10.50±0.07 | 1.47±0.00 | 7.94±0.03 |
| GRIN | 2.03±0.00 | 13.26±0.05 | 1.91±0.00 | 10.41±0.03 | 1.14±0.01 | 6.60±0.10 | 0.67±0.00 | 1.55±0.01 |
| V-RIN | 6.84±0.17 | 150.08±6.13 | 3.96±0.08 | 49.98±1.30 | 2.49±0.04 | 36.12±0.66 | 1.21±0.03 | 6.08±0.29 |
| GP-VAE | 6.55±0.09 | 122.33±2.05 | 6.57±0.10 | 127.26±3.97 | 2.86±0.15 | 26.80±2.10 | 3.41±0.23 | 38.95±4.16 |
| rGAIN | 2.90±0.01 | 21.67±0.15 | 2.83±0.01 | 20.03±0.09 | 2.18±0.01 | 13.96±0.20 | 1.88±0.02 | 10.37±0.20 |
| CSDI | 1.98±0.00 | 12.62±0.60 | 1.79±0.00 | 8.96±0.08 | 0.86±0.00 | 4.39±0.02 | 0.57±0.00 | 1.12±0.03 |
| SSSD | 2.95±0.01 | 23.48±0.09 | 2.83±0.02 | 21.95±0.14 | 1.03±0.01 | 7.32±0.05 | 0.97±0.01 | 2.98±0.03 |
| PriSTI | 1.86±0.00 | 10.70±0.02 | 1.72±0.00 | 8.24±0.05 | 0.78±0.00 | 3.31±0.01 | 0.55±0.00 | 1.03±0.00 |
| PriSTI† variant | 1.83±0.00 | 10.57±0.00 | 1.72±0.01 | 8.20±0.00 | 0.77±0.01 | 3.29±0.01 | 0.52±0.00 | 1.01±0.01 |
| **FastSTI-2 (6)** | 1.81±0.01 | 10.44±0.00 | 1.73±0.00 | 8.17±0.03 | 0.78±0.00 | 3.28±0.03 | 0.51±0.00 | 0.98±0.01 |
| **FastSTI-4 (6)** | **1.79±0.01** | **10.38±0.00** | **1.71±0.00** | **8.15±0.02** | **0.75±0.00** | **3.26±0.02** | **0.50±0.00** | **0.96±0.01** |

| | PEMS04 | | | | PEMS08 | | | |
|---|---|---|---|---|---|---|---|---|
| | Block-missing (10.59%) | | Point-missing (26.21%) | | Block-missing (9.41%) | | Point-missing (25.25%) | |
| | MAE | RMSE | MAE | RMSE | MAE | RMSE | MAE | RMSE |
| Mean | 40.89±0.00 | 83.41±0.00 | 39.46±0.00 | 77.97±0.00 | 37.54±0.00 | 76.14±0.00 | 35.11±0.00 | 70.28±0.00 |
| KNN | 39.28±0.00 | 75.41±0.00 | 35.14±0.00 | 66.90±0.00 | 33.49±0.00 | 70.25±0.00 | 31.06±0.00 | 63.41±0.00 |
| Lin-ITP | 27.74±0.00 | 55.41±0.00 | 23.46±0.00 | 51.54±0.00 | 25.79±0.00 | 53.87±0.00 | 22.01±0.00 | 40.77±0.00 |
| KF | 46.21±0.00 | 87.36±0.00 | 43.15±0.00 | 80.16±0.00 | 40.03±0.00 | 79.98±0.00 | 37.51±0.00 | 76.10±0.00 |
| MICE | 29.56±0.01 | 57.84±0.04 | 27.15±0.06 | 50.28±0.36 | 27.94±0.01 | 49.14±0.09 | 24.76±0.02 | 39.97±0.07 |
| VAR | 24.68±0.01 | 50.16±0.01 | 20.44±0.01 | 32.87±0.06 | 22.69±0.05 | 36.94±0.01 | 17.92±0.01 | 29.36±0.02 |
| TRMF | 21.72±0.01 | 34.12±0.01 | 18.37±0.01 | 30.97±0.06 | 20.32±0.05 | 35.74±0.01 | 15.32±0.01 | 26.99±0.02 |
| BATF | 27.10±0.13 | 40.28±0.01 | 20.89±0.01 | 34.11±0.02 | 23.87±0.05 | 37.88±0.03 | 18.26±0.03 | 29.21±0.02 |
| BRITS | 20.24±0.01 | 33.05±0.02 | 17.87±0.03 | 29.74±0.01 | 18.37±0.02 | 32.51±0.01 | 14.67±0.01 | 25.33±0.03 |
| GRIN | 18.77±0.03 | 29.88±0.08 | 16.94±0.01 | 26.11±0.01 | 12.98±0.02 | 24.23±0.10 | 11.37±0.01 | 18.21±0.01 |
| V-RIN | 25.78±0.01 | 49.11±0.01 | 24.26±0.01 | 43.57±0.24 | 24.56±0.01 | 40.13±0.22 | 22.77±0.02 | 35.49±0.08 |
| GP-VAE | 27.83±0.15 | 52.97±0.01 | 27.54±0.01 | 50.99±0.01 | 21.5±0.06 | 36.98±0.05 | 19.99±0.13 | 31.43±0.01 |
| rGAIN | 17.67±0.01 | 29.84±0.07 | 15.21±0.04 | 25.03±0.01 | 14.03±0.01 | 24.72±0.01 | 10.98±0.02 | 19.34±0.01 |
| CSDI | 16.58±0.01 | 28.13±0.02 | 15.09±0.01 | 24.91±0.02 | 12.06±0.05 | 23.08±0.10 | 10.07±0.01 | 17.19±0.00 |
| SSSD | 17.93±0.03 | 30.19±0.01 | 15.32±0.04 | 25.11±0.01 | 13.96±0.01 | 24.63±0.01 | 10.97±0.01 | 19.21±0.01 |
| PriSTI | 16.41±0.01 | 28.01±0.03 | 14.87±0.01 | 24.53±0.01 | 11.89±0.01 | 22.24±0.01 | 10.05±0.00 | 17.13±0.02 |
| PriSTI† variant | 16.36±0.00 | 27.93±0.00 | 14.75±0.00 | 24.50±0.00 | 11.76±0.01 | 22.10±0.00 | 9.94±0.00 | 17.03±0.00 |
| **FastSTI-2 (6)** | 16.28±0.00 | 27.89±0.01 | 14.69±0.01 | 24.47±0.03 | **11.60±0.01** | **21.79±0.01** | 9.89±0.00 | 16.97±0.01 |
| **FastSTI-4 (6)** | **16.21±0.01** | **27.70±0.00** | **14.67±0.02** | **24.42±0.01** | 11.73±0.01 | 22.00±0.00 | **9.80±0.01** | **16.93±0.01** |

- **FastSTI-4 ($K$=6) w/o acc**. FastSTI-4 with 50 reverse steps and $K$=6 of Diff-GCN.
- **FastSTI-2 ($K$=2) w/o acc**. FastSTI-2 with 50 reverse steps and $K$=2 of Diff-GCN.
- **FastSTI-4 ($K$=2) w/o acc**. FastSTI-4 with 50 reverse steps and $K$=2 of Diff-GCN.

2) *Acceleration*:

- **PriSTI w/ our acc**. PriSTI of with 6 reverse steps, integrated with *our acceleration method* (see Section IV-E).
- **FastSTI-2 ($K$=6) w/ acc**. FastSTI-2 with 6 reverse steps and $K$=6 of Diff-GCN.
- **FastSTI-4 ($K$=6) w/ acc**. FastSTI-4 with 6 reverse steps and $K$=6 of Diff-GCN.
- **FastSTI-2 ($K$=2) w/ acc**. FastSTI-2 with 6 reverse steps and $K$=2 of Diff-GCN.
- **FastSTI-4 ($K$=2) w/ acc**. FastSTI-4 with 6 reverse steps and $K$=2 of Diff-GCN.

Figures 5 and 6 display the imputation time of the models for all sensors throughout 24-time points (i.e., for 2 hours at 5-minute interval) on traffic speed and flow datasets. We can conclude that: 1) FastSTI utilizes a tuned variance schedule to speed up inference time in the imputation phase, which enables FastSTI to impute highly accurate traffic data with only 6 reverse steps ($8.3\times$ less than the 50 steps of both PriSTI [20] and CSDI [18]). Meanwhile, from Figure 5, our FastSTI-2 ($K$=2) w/ acc. ($\sim 12.10s$) is $6.5\times$ faster than PriSTI ($\sim 78.96s$) and $5\times$ faster than CSDI ($\sim 60.2s$) on METR-LA. With the higher-order FastSTI-4 w/ acc. ($\sim 31.44s$), the model requires more inference time; however, FastSTI-4 ($K$=2) w/

TABLE VI
CPRS COMPARISON ([**BOLD** = BEST, AND UNDERLINE = SECOND BEST]. (6) REPRESENTS THAT OUR ACCELERATED METHODS ONLY REQUIRES 6 REVERSE STEPS DURING THE INFERENCE.

| Method | METR-LA | | PEMS-BAY | | PEMS04 | | PEMS08 | |
|---|---|---|---|---|---|---|---|---|
| | Block-missing | Point-missing | Block-missing | Point-missing | Block-missing | Point-missing | Block-missing | Point-missing |
| V-RIN | 0.1283 | 0.7781 | 0.0394 | 0.0191 | 0.1842 | 0.1524 | 0.1522 | 0.1198 |
| GP-VAE | 0.1118 | 0.0977 | 0.0436 | 0.0568 | 0.1623 | 0.1358 | 0.1436 | 0.1103 |
| CSDI | 0.0260 | 0.0235 | 0.0127 | 0.0067 | 0.0583 | 0.0547 | 0.0420 | 0.0371 |
| PriSTI | 0.0244 | 0.0227 | **0.0093** | 0.0064 | 0.0556 | 0.0522 | 0.0418 | 0.0364 |
| **FastSTI-2 (6)** | 0.0243 | 0.0223 | 0.0095 | 0.0062 | 0.0542 | 0.0505 | **0.0373** | **0.0307** |
| **FastSTI-4 (6)** | **0.0241** | **0.0219** | **0.0093** | **0.0060** | **0.0540** | **0.0503** | 0.0374 | **0.0305** |

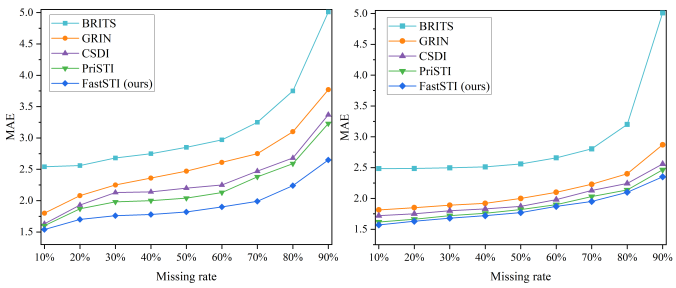



Fig. 3. The Impact of Missing Rate on the Imputation Performance on METR-LA Dataset.

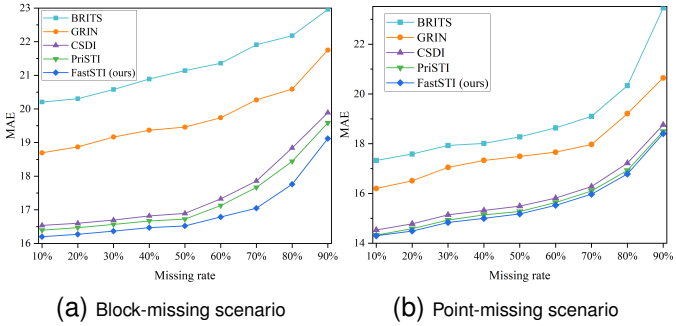



Fig. 4. The Impact of Missing Rate on the Imputation Performance on PEMS04 Dataset.

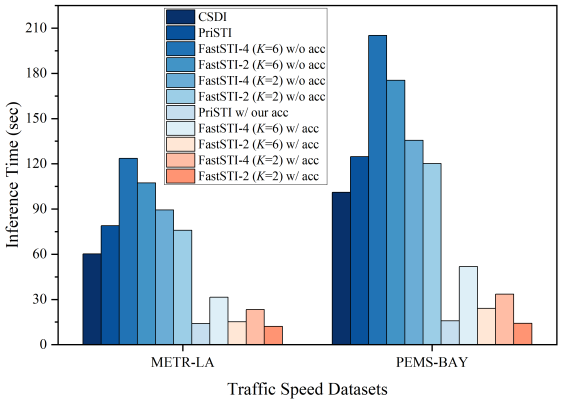



Fig. 5. Inference Times on Traffic Speed Datasets (METR-LA and PEMS-BAY).

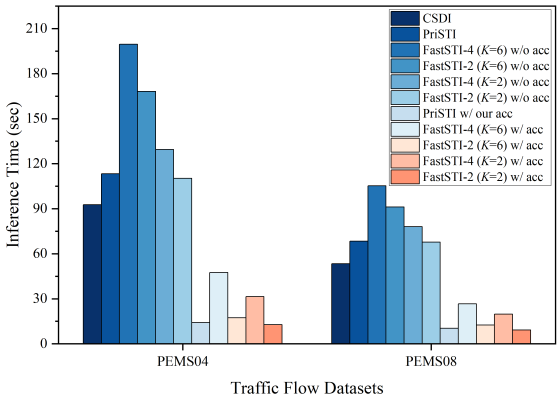



Fig. 6. Inference Times on Traffic Flow Datasets (PEMS04 and PEMS08).

acc is still 3.4× faster than PriSTI and 2.5x faster than CSDI. Thereby, FastSTI significantly reduces the computational time required compared to these competing diffusion architectures. 2) The larger the parameters of Diff-GCN, the slower the inference speed. As shown in Figure 6, with the accelerated version, FastSTI-2 ($K$=2) is 18.7 seconds faster than FastSTI-2 ($K$=6) at every two hours of imputation on the PEMS04 dataset. Similarly, FastSTI-2 ($K$=2) is 10.6 seconds faster than FastSTI-2 ($K$=6) on PEMS08. This indicates that increasing the $K$ parameter in Diff-GCN allows for more geospatial information extraction from traffic patterns, but it also adds to the computational time burden. 3) Our acceleration technique can be compatible with most diffusion-based models (e.g., PriSTI [20]). Compared to our FastSTI-2 ($K$=2), PriSTI can obtain a close inference time using our acceleration method.

### G. Ablation Study on Imputation Performance

We conduct an ablation study on the METR-LA datasets to verify the impact of different components of our FastSTI model. Table VII illustrates the performance of FastSTI-4 with and without employing our proposed components, including the pseudo-numerical method (PN) and the GCN-based conditional extractor (Diff-GCN). When we remove the pseudo-numerical (PN) methods, the model no longer considers high-order numerical methods converging to the exact solution when $\Delta t$ is closer to 0. Consequently, it no longer utilizes a larger iteration interval $\Delta t$ to achieve global error reduction, resulting in decreased imputation accuracy. Additionally, the spatial learning sub-component, Diff-GCN, is crucial in extracting geographic interactions among nodes

TABLE VII
THE INFLUENCE OF DIFFERENT COMPONENTS ON FASTSTI-4.

| PN | Diff-GCN | METR-LA | | | |
|---|---|---|---|---|---|
| | | Block-missing | | Point-missing | |
| | | MAE | MSE | MAE | MSE |
| ✗ | ✗ | 1.86±0.00 | 10.70±0.02 | 1.72±0.00 | 8.24±0.05 |
| ✗ | ✓ | 1.83±0.00 | 10.40±0.00 | 1.74±0.00 | 8.22±0.00 |
| ✓ | ✗ | 1.89±0.00 | 10.87±0.00 | 1.82±0.00 | 8.57±0.00 |
| ✓ | ✓ | **1.79±0.01** | **10.38±0.00** | **1.71±0.00** | **8.15±0.02** |

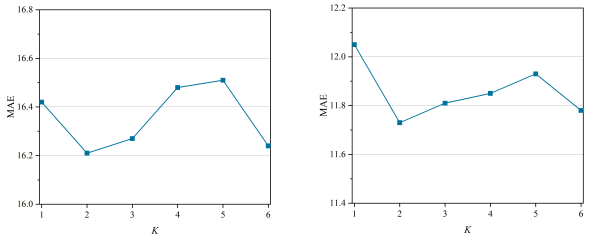
within the spatial correlation. When removing Diff-GCN, the model has a weak ability to capture the influence among nodes by spreading the traffic feature information on graph $G$. Therefore, integrating PN and Diff-GCN together into the model achieves the best performance.

For further ablation, we investigate the impact of increasing the number of denoising steps on our model. Table VIII reports a comparison between our FastSTI (6 steps) and FastSTI *w/o acceleration* when increasing the number of steps to 50. Surly, increasing the number of steps increases the performance by a noticeable gap, proving the effectiveness of FastSTI quality-wise. However, increasing the number of steps results in a dramatic increase in imputation time.

### H. Hyperparameter Analysis

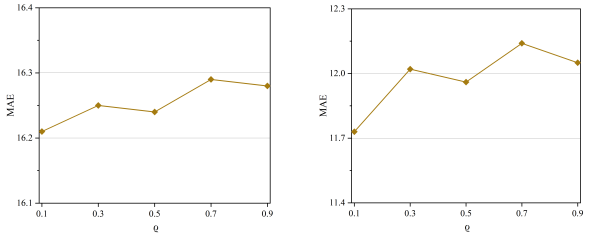
In this part, we explore the impact of FastSTI's diffusion parameters on the METR-LA dataset, including the minimum noise level $\beta_1$, the maximum noise level $\beta_T$, and the diffusion schedule to generate $\beta$. Table IX reveals that FastSTI performs best when setting $\beta_1$ to 0.0001, $\beta_T$ to 0.2, and utilizing a quadratic diffusion schedule. Here, the noise level parameter is employed to regulate the diffusion speed. The convergence speed of the model slows down when the $\beta_T$ is smaller, while the numerical stability of the model is compromised when the $\beta_T$ is larger. Additionally, regarding the diffusion schedule, a quadratic schedule outperforms linear and cosine schedules. This is because the quadratic schedule allows for a gentle decay of $\alpha_t$, which in turn improves sample quality, making it the optimal choice for FastSTI.

To further investigate our Diff-GCN feature extractor for capturing local geospatial dependencies, we report different parameters of graph random walk step $K$ and the graph coefficient $\varrho$ of FastSTI-4 on the PEMS04 and PEMS08 datasets under the block-missing scenario. 1) The parameter $K$ corresponds to the convolutional filter's receptive field size. Large $K$ values enable the module to capture a wide range of spatial graph information but come with the trade-off of increased learning complexity. As shown in Figures 7a and 7b, we can observe that with increasing the number of $K$, the error on the testing set initially decreases, followed by a slight increase. Compared with $K$=2, $K$=6 reports similar MAE values but heavily increases the time consumption. We also analyze the time performance of the setting of these two parameters in Sec. V-F. Therefore, $K$=2 can be selected as the optimal parameter. That is, the graph convolution operation can aggregate the features of the node itself, as well as those of its 1-hop and 2-hop graph neighbors when $K$=2. 2) The




(a) PEMS04 Block-missing (b) PEMS08 Block-missing

Fig. 7. The Effect of Diff-GCN's $K$ Parameter on Imputation Performance.




(a) PEMS04 Block-missing (b) PEMS08 Block-missing

Fig. 8. The Effect of Diff-GCN's $\varrho$ Parameter on Imputation Performance.

graph coefficient $\varrho$ represents the scaling factor controlling the influence of the neighboring nodes' matrix (excluding self-node) during feature propagation in the Diff-GCN. For road traffic pattern, where the self-node spatial correlation typically holds greater significance than neighboring nodes' contributions, setting the optimal $\varrho$=0.1 (see Fig. 8) means that the influence of $K$-hop (where $K > 0$) node's adjacency matrix on the feature propagation is scaled to 10% of their original value. Such configuration enables the model to primarily focus on a self-node adjacency matrix for feature propagation while slightly considering the impact of the neighboring nodes.

### I. Case Study

We present the visualized results of the case study on the METR-LA dataset; comparing METR-LA to PEMS-BAY (Bay Area) shows that traffic imputation on the METR-LA dataset (Los Angeles, famous for its complicated traffic conditions) is a more complicated way. Taking the block-missing scenario as an example, Fig. 9 provides a visualization of the two-hour imputation (24-time points) for five closely located sensors/nodes. Here, the green lines represent the imputed values generated by our FastSTI-4. The purple points indicate the ground-truth imputation targets. The black crosses represent the observed values, while the green shade is the quantiles between 5% and 95%. As the figure shows, sensors #1 and #4 exhibit minimal instances of missing data, while sensors #3 and #5 show a continuous pattern of missing values spanning approximately an hour. Moreover, in more extreme cases, i.e., sensor #2, there is a complete absence of usable observed value for two hours. It is also apparent that our FastSTI-4 shows positive results in most cases. For instance, sensor #5 can utilize its temporal feature observations and leverage the spatial correlation features from neighboring sensors to predict missing values. Indeed, it is crucial to emphasize that sensor

TABLE VIII
THE IMPACT OF INCREASING THE NUMBER OF STEPS ON FASTSTI . (6) REPRESENTS THAT FASTSTI ONLY REQUIRES 6 REVERSE STEPS *WITH* OUR ACCELERATED METHOD, AND (50) REPRESENTS FASTSTI USE 50 REVERSE STEPS *WITHOUT* ACCELERATED METHOD.

| Method | METR-LA | | | | | | PEMS-BAY | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Block-missing (16.52%) | | | Point-missing (31.09%) | | | Block-missing (9.20%) | | | Point-missing (25.01%) | | |
| | MAE | MSE | CRPS | MAE | MSE | CRPS | MAE | MSE | CRPS | MAE | MSE | CRPS |
| **FastSTI-2 (50)** | 1.80±0.00 | 10.34±0.01 | 0.0241 | 1.69±0.03 | 8.01±0.03 | 0.0220 | 0.72±0.01 | 3.26±0.00 | 0.0091 | 0.52±0.01 | 1.01±0.02 | 0.0061 |
| **FastSTI-4 (50)** | **1.79±0.01** | **10.29±0.00** | **0.0239** | **1.68±0.00** | **7.99±0.01** | **0.0219** | **0.70±0.00** | **3.24±0.00** | **0.0089** | 0.51±0.00 | 0.99±0.00 | **0.0059** |
| **FastSTI-2 (6)** | 1.81±0.01 | 10.44±0.00 | 0.0243 | 1.73±0.00 | 8.17±0.03 | 0.0223 | 0.78±0.00 | 3.28±0.03 | 0.0095 | 0.51±0.00 | 0.98±0.01 | 0.0062 |
| **FastSTI-4 (6)** | **1.79±0.01** | 10.38±0.00 | 0.0241 | 1.71±0.00 | 8.15±0.02 | **0.0219** | 0.75±0.00 | 3.26±0.02 | 0.0093 | **0.50±0.00** | **0.96±0.01** | 0.0060 |

| Method | PEMS04 | | | | | | PEMS08 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Block-missing (10.59%) | | | Point-missing (26.21%) | | | Block-missing (9.41%) | | | Point-missing (25.25%) | | |
| | MAE | RMSE | CRPS | MAE | RMSE | CRPS | MAE | RMSE | CRPS | MAE | RMSE | CRPS |
| **FastSTI-2 (50)** | 16.22±0.00 | 27.84±0.01 | 0.0539 | 14.64±0.01 | 24.42±0.03 | 0.0503 | **11.57±0.01** | **21.77±0.02** | **0.0371** | 9.82±0.02 | 16.93±0.01 | **0.0302** |
| **FastSTI-4 (50)** | **16.17±0.01** | **27.64±0.03** | **0.0537** | **14.63±0.01** | **24.38±0.01** | **0.0501** | 11.69±0.01 | 21.95±0.02 | 0.0373 | **9.74±0.01** | **16.89±0.01** | **0.0302** |
| **FastSTI-2 (6)** | 16.28±0.00 | 27.89±0.01 | 0.0542 | 14.69±0.01 | 24.47±0.03 | 0.0505 | 11.60±0.01 | 21.79±0.01 | 0.0373 | 9.89±0.00 | 16.97±0.01 | 0.0307 |
| **FastSTI-4 (6)** | 16.21±0.01 | 27.70±0.00 | 0.0540 | 14.67±0.02 | 24.42±0.01 | 0.0503 | 11.73±0.01 | 22.00±0.00 | 0.0374 | 9.80±0.01 | 16.93±0.01 | 0.0305 |

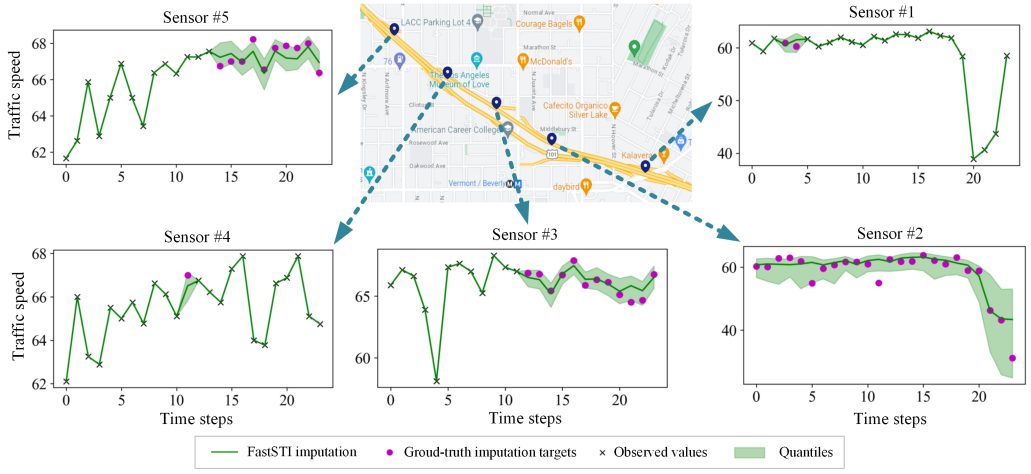



Fig. 9. Case study on METR-LA.

TABLE IX
INFLUENCE OF DIFFERENT DIFFUSION PARAMETERS ON METR-LA DATASET [**BOLD** = BEST].

| Diffsion parameters | | | Performance | |
|---|---|---|---|---|
| $\beta_1$ | $\beta_T$ | schedule | MAE | MSE |
| 0.0001 | 0.2 | linear | 1.96 | 12.46 |
| | | cosine | 1.88 | 11.78 |
| | | quadratic | **1.79** | **10.38** |
| 0.001 | 0.2 | quadratic | 2.06 | 13.87 |
| 0.0001 | 0.1 | linear | 1.95 | 12.37 |
| | 0.1 | quadratic | 1.86 | 11.54 |
| | 0.3 | quadratic | 1.91 | 11.89 |

#2, despite lacking observed values for two hours and thus not having access to its temporal correlation information, can still rely on observed data from spatial neighboring sensors to estimate the missing values. This highlights the capability of our FastSTI to achieve relatively satisfactory imputation results.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we investigate the challenges of spatiotemporal traffic data imputation in real-life scenarios, proposing our FastSTI model. FastSTI utilizes a higher-order pseudo-numerical methodology for a conditional diffusion model to enhance imputation accuracy. Furthermore, we demonstrate the effectiveness of incorporating conditional information through GCN variants (Diff-GCN) to serve as feature prior knowledge, capturing the spatiotemporal correlations. To accelerate the imputation process and fit real-life applications, we introduce the utilization of a variance schedule to reduce the sampling iterations of the diffusion model. The effectiveness of the proposed FastSTI in addressing real-world data missing scenarios is substantiated by experiments, showcasing its accelerated imputation time and competitive performance in imputation tasks.

In FastSTI, our primary focus is on a single feature attribute: traffic speed or flow. Traffic conditions are also influenced by various other factors, such as weather, Points of Interest (POI), and unexpected events. More attributes can be considered in the process of missing data estimation. Exploring this idea further could be a notable topic for future researchers.

## ACKNOWLEDGMENTS

This work was partially supported by the China Scholarship Council (No. 202206290090), by an UniPD BIRD-2021 Project, and by the PRIN-17 PREVUE project from the Italian MUR (CUP E94I19000650001). The authors would like to thank the reviewers for their helpful comments.

## REFERENCES


[1] X. Chen, M. Lei, N. Saunier, and L. Sun, "Low-rank autoregressive tensor completion for spatiotemporal traffic data imputation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12 301–12 310, 2021.

[2] M. Lei, A. Labbe, Y. Wu, and L. Sun, "Bayesian kernelized matrix factorization for spatiotemporal traffic data imputation and kriging," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 962–18 974, 2022.

[3] A. B. Said and A. Erradi, "Spatiotemporal tensor completion for improved urban traffic imputation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6836–6849, 2021.

[4] I. Laña, I. I. Olabarrieta, M. Vélez, and J. Del Ser, "On the imputation of missing data for road traffic forecasting: New insights and novel techniques," *Transportation research part C: emerging technologies*, vol. 90, pp. 18–33, 2018.

[5] L. Beretta and A. Santaniello, "Nearest neighbor imputation algorithms: a critical evaluation," *BMC medical informatics and decision making*, vol. 16, no. 3, pp. 197–208, 2016.

[6] J. Kihoro, K. Athiany *et al.*, "Imputation of incomplete nonstationary seasonal time series data," *Mathematical Theory and Modeling*, vol. 3, no. 12, pp. 142–154, 2013.

[7] I. R. White, P. Royston, and A. M. Wood, "Multiple imputation using chained equations: issues and guidance for practice," *Statistics in medicine*, vol. 30, no. 4, pp. 377–399, 2011.

[8] Q. Shang, Z. Yang, S. Gao, and D. Tan, "An imputation method for missing traffic data based on fcm optimized by pso-svr," *Journal of Advanced Transportation*, vol. 2018, pp. 1–21, 2018.

[9] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "Brits: Bidirectional recurrent imputation for time series," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[10] A. Cini, I. Marisca, and C. Alippi, "Filling the g_ap_s: Multivariate time series imputation by graph neural networks," *International Conference on Learning Representations*, 2022.

[11] W. Zhang, P. Zhang, Y. Yu, X. Li, S. A. Biancardo, and J. Zhang, "Missing data repairs for traffic flow with self-attention generative adversarial imputation net," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7919–7930, 2021.

[12] A. Bertugli, S. Calderara, P. Coscia, L. Ballan, and R. Cucchiara, "AC-VRNN: Attentive Conditional-VRNN for multi-future trajectory prediction," *Computer Vision and Image Understanding*, 2021.

[13] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[14] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *International Conference on Learning Representations*, 2021.

[15] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8162–8171.

[16] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, Y. Shao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *arXiv preprint arXiv:2209.00796*, 2022.

[17] L. Lin, Z. Li, R. Li, X. Li, and J. Gao, "Diffusion models for time series applications: A survey," *arXiv preprint arXiv:2305.00624*, 2023.

[18] Y. Tashiro, J. Song, Y. Song, and S. Ermon, "Csdi: Conditional score-based diffusion models for probabilistic time series imputation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 804–24 816, 2021.

[19] J. M. L. Alcaraz and N. Strodthoff, "Diffusion-based time series imputation and forecasting with structured state space models," *Transactions on Machine Learning Research*, 2022.

[20] M. Liu, H. Huang, H. Feng, L. Sun, B. Du, and Y. Fu, "Pristi: A conditional diffusion framework for spatiotemporal imputation," *International Conference on Data Engineering*, 2023.

[21] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," *Advances in neural information processing systems*, vol. 29, 2016.

[22] X. Chen, Z. He, Y. Chen, Y. Lu, and J. Wang, "Missing traffic data imputation and pattern discovery with a bayesian augmented tensor factorization model," *Transportation Research Part C: Emerging Technologies*, vol. 104, pp. 66–77, 2019.

[23] J. Ming, L. Zhang, W. Fan, W. Zhang, Y. Mei, W. Ling, and H. Xiong, "Multi-graph convolutional recurrent network for fine-grained lane-level traffic flow imputation," in *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2022, pp. 348–357.

[24] V. Fortuin, D. Baranchuk, G. Rätsch, and S. Mandt, "Gp-vae: Deep probabilistic time series imputation," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 1651–1661.

[25] H. Qin, X. Zhan, Y. Li, X. Yang, and Y. Zheng, "Network-wide traffic states imputation using self-interested coalitional learning," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 1370–1378.

[26] P.-A. Mattei and J. Frellsen, "Miwae: Deep generative modelling and imputation of incomplete data sets," in *International conference on machine learning*. PMLR, 2019, pp. 4413–4423.

[27] J. Yoon, J. Jordon, and M. Schaar, "Gain: Missing data imputation using generative adversarial nets," in *International conference on machine learning*. PMLR, 2018, pp. 5689–5698.

[28] Y. Yuan, Y. Zhang, B. Wang, Y. Peng, Y. Hu, and B. Yin, "Stgan: Spatio-temporal generative adversarial network for traffic data imputation," *IEEE Transactions on Big Data*, vol. 9, no. 1, pp. 200–211, 2022.

[29] J. Chen, S. Zhang, X. Chen, Q. Jiang, H. Huang, and C. Gu, "Learning traffic as videos: a spatio-temporal vae approach for traffic data imputation," in *International Conference on Artificial Neural Networks*. Springer, 2021, pp. 615–627.

[30] T. W. Richardson, W. Wu, L. Lin, B. Xu, and E. A. Bernal, "Mcflow: Monte carlo flow models for data imputation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 205–14 214.

[31] E. A. Brempong, S. Kornblith, T. Chen, N. Parmar, M. Minderer, and M. Norouzi, "Denoising pretraining for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4175–4186.

[32] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg, "Structured denoising diffusion models in discrete state-spaces," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 981–17 993, 2021.

[33] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

[34] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *International Conference on Learning Representations*, 2021.

[35] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," *Advances in Neural Information Processing Systems*, 2022.

[36] G. Wanner and E. Hairer, *Solving ordinary differential equations II*. Springer Berlin Heidelberg New York, 1996, vol. 375.

[37] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," *International Conference on Learning Representations*, 2021.

[38] L. Liu, Y. Ren, Z. Lin, and Z. Zhao, "Pseudo numerical methods for diffusion models on manifolds," *International Conference on Learning Representations*, 2022.

[39] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[40] B. Yang, Y. Kang, Y. Yuan, X. Huang, and H. Li, "St-lbagan: Spatio-temporal learnable bidirectional attention generative adversarial networks for missing traffic data imputation," *Knowledge-Based Systems*, vol. 215, p. 106705, 2021.

[41] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *International Conference on Learning Representations*, 2018.

[42] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," *arXiv preprint arXiv:2009.09761*, 2020.

[43] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 922–929.

[44] J. E. Matheson and R. L. Winkler, "Scoring rules for continuous probability distributions," *Management science*, vol. 22, no. 10, pp. 1087–1096, 1976.

[45] A. W. Mulyadi, E. Jun, and H.-I. Suk, "Uncertainty-aware variational-recurrent imputation network for clinical time series," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9684–9694, 2021.

**Shaokang Cheng** is currently pursuing his Ph.D. degree in Transportation Engineering at the School of Automation, Northwestern Polytechnical University, China. He is also a visiting Ph.D. student with the Department of Mathematics "T. Levi-Civita", University of Padova, Italy. His research interests mainly include spatiotemporal data mining, urban computing, and deep learning.

**Nada Osman** received her B.S. degree in computer engineering from Computer and Systems Engineering Department at Alexandria University, Egypt, in 2016. From 2016 to 2018, she worked as a software engineer at Ejada Systems Ltd., Alexandria, Egypt. From 2018 to 2020, she worked as a research assistant in the CRC lab and as a teaching assistant in Computer and Systems Engineering Department at Alexandria University, Egypt. She received her M.S. degree in computer engineering from the same university in 2020. She is currently pursuing her Ph.D. at the Department of Mathematics, University of Padova, Italy. Her research interests include computer vision, signal processing, AI, machine learning, and deep learning.

**Shiru Qu** is a Full Professor at the School of Automation, Northwestern Polytechnical University, China. She received her Ph.D. degree in Automatic Control from Northwestern Polytechnical University in 2002. In the same university, she has been a university researcher since 1984 and an Associate Professor since 1995. Her research focuses on intersection of machine learning, data mining, computer vision, and intelligent transportation systems.

**Lamberto Ballan** is an Associate Professor of computer science in the Department of Mathematics "Tullio Levi-Civita" at the University of Padova, Italy, where he leads the Visual Intelligence and Machine Perception (VIMP) group. He is also affiliated faculty at the Human Inspired Technology Research Centre. Previously, he was a senior postdoctoral researcher at Stanford University and University of Florence, Italy, supported by a prestigious Marie Curie Fellowship from the European Commission. He received his M.S. and Ph.D. degrees in computer engineering in 2006 and 2011, both from the Univ. of Florence. His primary research area is computer vision, closely integrated with machine learning and multimedia. He received the IVC best paper award 2021, and he has been an ELLIS member since 2021.