Networked Time-series Prediction with Incomplete Data via Generative Adversarial Network

YICHEN ZHU, BO JIANG, HAIMING JIN, and MENGTIAN ZHANG, Shanghai Jiao Tong University, Shanghai, China
FENG GAO, Zhejiang Lab, Hangzhou, China
JIANQIANG HUANG, Alibaba Damo Academy, Hangzhou, China
TAO LIN, Communication University of China, Beijing, China
XINBING WANG, Shanghai Jiao Tong University, Shanghai, China

A *networked time series (NETS)* is a family of time series on a given graph, one for each node. It has a wide range of applications from intelligent transportation to environment monitoring to smart grid management. An important task in such applications is to predict the future values of a NETS based on its historical values and the underlying graph. Most existing methods require complete data for training. However, in real-world scenarios, it is not uncommon to have missing data due to sensor malfunction, incomplete sensing coverage, and so on. In this article, we study the problem of *NETS prediction with incomplete data*. We propose networked time series Imputation Generative Adversarial Network (NETS-ImpGAN), a novel deep learning framework that can be trained on incomplete data with missing values in both history and future. Furthermore, we propose *Graph Temporal Attention Networks*, which incorporate the attention mechanism to capture both inter-time series and temporal correlations. We conduct extensive experiments on four real-world datasets under different missing patterns and missing rates. The experimental results show that NETS-ImpGAN outperforms existing methods, reducing the Mean Absolute Error by up to 25%.

CCS Concepts: • **Information systems** → **Data mining**; • **Computing methodologies** → **Neural networks;**

Additional Key Words and Phrases: Networked time series, incomplete data, prediction, imputation

ACM Reference Format:
Yichen Zhu, Bo Jiang, Haiming Jin, Mengtian Zhang, Feng Gao, Jianqiang Huang, Tao Lin, and Xinbing Wang. 2024. Networked Time-series Prediction with Incomplete Data via Generative Adversarial Network. *ACM Trans. Knowl. Discov. Data.* 18, 5, Article 115 (February 2024), 25 pages. https://doi.org/10.1145/3643822

This work is supported in part by National Natural Science Foundation of China (Grant No. 62072302, 62372288, U20A20181, U22A6001, 61960206002, 62262018), Alibaba Innovative Research (AIR) programme, and the Open Research Project of the State Key Laboratory of Media Convergence and Communication, Communication University of China, China (No.SKLMCC2021KF011).
Authors' addresses: Y. Zhu, B. Jiang (Corresponding author), H. Jin, M. Zhang, and X. Wang, Shanghai Jiao Tong University, Shanghai, China; e-mails: zyc_ieee@sjtu.edu.cn, bjiang@sjtu.edu.cn, jinhaiming@sjtu.edu.cn, zhangmengtian@sjtu.edu.cn, xwang8@sjtu.edu.cn; F. Gao, Zhejiang Lab, Hangzhou, China; e-mail: gaof@zhejianglab.com; J. Huang, Alibaba Damo Academy, Hangzhou, China; e-mail: jianqiang.jqh@gmail.com; T. Lin, Communication University of China, Beijing, China; e-mail: lintao@cuc.edu.cn.

ACM 1556-4681/2024/02-ART115
https://doi.org/10.1145/3643822

1 INTRODUCTION

A **networked time series (NETS)** is a family of time series on a given graph, where each node is associated with a time series [8]. Depending on the application, the underlying graph may encode spatial proximity, statistical dependency, or other contextual or structural information about the time series. As a versatile modeling tool, NETS has a wide range of applications from intelligent transportation, environment monitoring to smart grid management.

An important task in such applications is to predict the future values of a NETS based on its historical values and the underlying graph. This has been studied extensively, and many prediction methods have been proposed in various contexts; see References [2, 6, 8, 12, 17, 23, 29, 33, 44, 48, 55, 57, 59, 60, 62, 64–66] and references therein. Most of these methods use deep learning and require complete data for training [2, 6, 17, 23, 29, 33, 44, 48, 55, 57, 59, 60, 62, 64–66]. They typically use GCN [26] to capture inter-time-series correlations and variants of CNN [54] or RNN [46] to capture temporal correlations. However, in real-world scenarios, it is not uncommon to have missing data due to sensor malfunction, incomplete sensing coverage, and so on. Simply removing all samples with missing data could lead to low data efficiency as demonstrated later in Section 5.5, since some observed data will also be removed [3]. First imputing the incomplete data and then predicting with the imputed data can result in error accumulation, as shown in Section 5.3. This motivates us to study the problem of *NETS prediction with incomplete data*.

Several deep learning methods [2, 29, 55, 56, 66] can predict from incomplete history data. They take incomplete history as input and compute reconstructionloss on the complete predicted future. However, they require complete future data for supervision during training and do not provide a full solution to our problem.

DCMF [8] and GRIN [12] are two NETS imputation methods that do not require complete data for training. They can be used for prediction by treating the future as missing data [9]. DCMF is always fitted to a single sample, so it cannot benefit from training on multiple samples to learn more complex temporal dependencies than allowed by its assumed linear system model. GRIN can be trained on multiple samples; however, it performs supervision only through the reconstruction loss on the observed values, with no direct control on the quality of the more important missing part. In addition, its bidirectional architecture may not be a perfect match for prediction, since there is no information passing from future to past.

MisGAN [31] is a general distribution learning framework that can supervise the missing part with incomplete data. It can be adapted for NETS prediction by plugging in properly designed inner modules. The framework first learns the joint distribution of the complete history and future, and then uses the learned complete data distribution to supervise prediction. As later shown in Section 5.4.2, the imperfectly learned complete data distribution can lead to significant error accumulation.

To overcome the aforementioned problems, we present **networked time series Imputation Generative Adversarial Network (NETS-ImpGAN)**, a novel deep learning framework for NETS prediction with incomplete data. NETS-ImpGAN consists of an outer framework called **Imputation GAN (ImpGAN)**, and a collection of inner modules called **Graph Temporal Attention Networks (GTANs)**.

The outer framework ImpGAN is a **Generative Adversarial Network (GAN)** [19] for imputation. Conceptually, we can regard prediction as a special type of imputation as in Reference [9], where the future is considered as missing. ImpGAN aims to learn the conditional distribution of missing values given observed values. By supervision on distributions rather than on individual values as in GRIN, we can more readily use observed values in different incomplete samples to guide the prediction. More specifically, ImpGAN has a generator that takes incomplete history and noise as input and outputs complete samples including predicted future and completed history.

The complete samples are then properly masked to generate new fake incomplete samples, which the discriminator tries to distinguish from real incomplete samples. As such, ImpGAN is a generic framework for imputation and may be of independent interest (Section 5.3.1). Note that real incomplete samples are used directly to guide the imputation process, which largely avoids the error accumulation in MisGAN.

The inner GTAN modules specialize ImpGAN to NETS prediction. As the implementations of the generators and discriminators of ImpGAN, GTANs are designed to properly capture inter-time-series correlations and temporal correlations. More specifically, Graph Attention Network [53] is used to capture inter-time-series correlations, and Multi-Head Self-Attention [52] and CNN [54] are used to capture temporal correlations. We follow the common practice of filling missing values with random noise or constants, so that all samples have the same shape. As noted in Reference [66], this may lead to inferior performance due to error accumulation. The incorporated attention mechanisms can help mitigate error accumulation by differentiating the observed values and filled values. Note that GTANs can also be used as a stand-alone model for NETS prediction with complete data (Section 5.3.2) or plugged into other frameworks such as MisGAN (Section 5.4.2).

To summarize, we make the following contributions.

— We propose NETS-ImpGAN, a novel deep learning framework for NETS prediction with incomplete data. The proposed framework can capture the complex dependencies from history to future with data that has missing values in both history and future.
— We propose GTANs to capture the inter-time-series correlations and temporal correlations of incomplete NETS. GTANs mitigate error accumulation by incorporating attention mechanisms.
— We conduct experiments on four real-world datasets under different missing patterns and missing rates. The results show that NETS-ImpGAN outperforms existing methods with up to 25% reduction in prediction error.

The rest of the article is organized as follows. Section 2 reviews the related work. Section 3 formulates the problem. Section 4 presents the NETS-ImpGAN framework, followed by evaluation in Section 5. Section 6 concludes the article.

2 RELATED WORK

NETS prediction. The problem of NETS prediction has been studied extensively in various contexts. Early works [64, 65] capture inter-time-series correlations with CNN [54] and temporal correlations by aggregating different timestamps with linear weighted sum. Some works [59, 60] then use LSTM [22] to capture non-linear temporal correlations. All these works use CNN and are limited to grid-structured graph. Many later works use GCN [26] or its variants for the general graph setting. We only introduce the state-of-the-art methods below. DCRNN [33] proposes diffusion-based GCN for inter-time-series correlations and uses GRU [11] for temporal correlations. STGCN [62] combines GCN and GLU [13] to capture both inter-time-series and temporal correlations. Graph WaveNet [57] captures inter-time-series correlations by proposing a variant of GCN with adaptive adjacency matrix and uses WaveNet [51] to capture temporal correlations. LSGCN [23] uses a variant of GCN with gated mechanism and GLU to capture inter-time-series and temporal correlations, respectively. Some other works, for example [6, 17, 44, 48], additionally incorporate periodic or scenario-specific auxiliary information to improve the performance. The prediction of tensor-valued NETS has also been considered in Jing et al. [24], which proposes Tensor GCN and Tensor RNN. However, all these methods require complete history data as input.

Some works on NETS prediction can take incomplete history data as input. STGNN-DAE [29] combines GCN and GLU in a denoising autoencoder for imputation. IGNNK [56] generates

random subgraphs and uses Diffusion GCN [33] to learn the spatial message passing mechanism for imputation. STGNN-DAE and IGNNK can be used for prediction by treating the future as missing data. SSSDS4 [2] uses a generative framework based on conditional diffusion [28] and incorporates structured state space model [20] for long-term temporal correlations. RIHGCN [66] is a deep learning framework for NETS prediction. It first uses GCN and LSTM to impute missing data in the history and then uses the completed history to predict the future. The imputation and prediction steps are trained jointly to mitigate error accumulation. RIHGCN also uses multiple graphs, static and dynamic, to capture dynamic inter-time-series correlation. GSTAE [55] regard imputation and prediction as parallel tasks and train them sequentially to mitigate error accumulation. It combines GCN and GRU in an autoencoder for NETS prediction. However, all these methods require complete future data for training.

There is limited literature on NETS prediction that allows incomplete input and does not require complete data for training. DCMF [8] is a matrix factorization method for missing data imputation of NETS. Its key assumptions are that a NETS and its associated graph have certain low-rank matrix representations and that the temporal dynamics is described by a first-order linear system. Facets [9] extends DCMF to tensor-valued NETS by using tensor decomposition instead of matrix factorization. NetDyna [21] further considers the case where the underlying graph is also incomplete. All three models are fitted to a single sample with no future data. Without explicit learning to predict the future, their predictive power relies critically on the strong and potentially restrictive assumption of linear system model. S-MKKM [18] incorporates a spatial multi-kernel clustering method into adaptive-weight non-negative matrix factorization for imputation of NETS. WDGTC [34] considers so-called weakly dependent modes in tensor completion. SD-ADMM [41] decomposes a vector of time series into components with different characteristics, such as smooth, periodic, nonnegative, or sparse. Without a model for temporal dynamics, S-MKKM, WDGTC, and SD-ADMM lack the ability to predict the future. GRIN [12] combines bidirectional RNN [47] with GCN for NETS imputation. It computes reconstruction loss only on the observed part of the samples and has no direct supervision on the missing part. SPIN [39] uses attention mechanism along both the graph and temporal dimensions for NETS imputation, but it requires additional auxiliary information of temporal features and geographic location, which is not available in our problem.

In what follows, we also review the imputation methods for other data types that does not require complete data for training.

Multiple time-series imputation. There are many methods for multiple time-series imputation in the literature. They can also be used for prediction by treating the future as missing data. Among them, TRMF [63], BRITS [10], E^{2}GAN [36, 37], and CSDI [49] can use incomplete data for training. TRMF is also natively proposed for the prediction task, and it combines a novel regularization scheme along the temporal dimension with matrix factorization. BRITS imputes missing values using a bidirectional RNN with a temporal decay factor. It minimizes the reconstruction errors on observed values and forces consistency between the imputed values in both directions. E^{2}GAN is a GAN-based [19] framework where the generator imputes missing values and the discriminator tries to distinguish imputed samples from samples with constant filled missing values. To capture temporal correlations of incomplete data, the generator and discriminator use a novel GRUI cell, which incorporates a temporal decay factor into GRU. CSDI uses a generative framework based on conditional diffusion [2] and incorporates Transformer layers [52] along both the temporal and feature dimensions. mSSA [1] uses low-order polynomials for trends, finite sum of harmonics for seasonality and linear time-invariant systems for imputation and prediction of incomplete multiple time series. All these methods take into account the correlations between all the time series in the imputation process, effectively treating multiple time series as a special

NETS with a complete underlying graph. As such, they cannot exploit the additional information provided by the graph of a general NETS.

General data imputation. There are several imputation methods for general data. GAIN [61] is an adaptation of GAN, where the generator imputes the missing entries, and the discriminator distinguishes between the observed and missing entries. MisGAN [31] learns the complete data distribution from incomplete data and uses it to supervise the missing entries in imputation. Partial VAE [38], MIWAE [40], and P-BiGAN [32] extend VAE [25], IWAE [7], and BiGAN [16], respectively, to learn the prior and posterior distributions of incomplete data given its latent representation and mask. They maximizes the likelihood of the observed entries and have no direct control on the missing entries. MIRACLE [30] is proposed as a regularization scheme that encourages the imputation to be consistent with the causal structure of data. It needs to be used jointly with other imputation methods as a refinement. GENIE [15] is proposed as a denoising solver for the diffusion-based data perturbation process. Though the data perturbation process may not be consistent with the actual missing pattern of incomplete data, it still has the potential to be used for imputation. All these methods except GENIE can use incomplete data for training. These methods are evaluated on images with two-dimensional (2D) CNN-based models to capture spatial correlations of grid-structured data and cannot be directly applied to general NETS. We adapt MisGAN to NETS and compare with it in Section 5.4.2.

Compared to existing methods, the proposed NETS-ImpGAN can directly supervise the missing part with only incomplete future data and adapt to different incomplete samples when capturing both the inter-time-series correlations and temporal correlations of NETS.

3 PROBLEM FORMULATION

3.1 Background and Notation

A NETS is a family of time series defined on a given graph. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with node set $\mathcal{V} = \{1, 2, \ldots, V\}$ and edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. A NETS on $\mathcal{G}$ over timestamps $\mathcal{T} = \{1, 2, \ldots, T\}$ is $(\mathbf{X}, \mathcal{G})$, where $\mathbf{X} = (X_{v,t}) \in \mathbb{R}^{V \times T}$ is a matrix whose entry $X_{v,t}$ is the value at timestamp $t \in \mathcal{T}$ of the time series on node $v \in \mathcal{V}$. Since we consider the case where the graph $\mathcal{G}$ is fixed and known, we will refer to a NETS by $\mathbf{X}$ for simplicity. It is understood that the underlying graph $\mathcal{G}$ is given.

In the presence of missing data, only part of $\mathbf{X}$ is observed. A binary mask $\mathbf{M} = (M_{v,t}) \in \{0, 1\}^{V \times T}$ indicates which entries of $\mathbf{X}$ are observed: $M_{v,t} = 1$ if $X_{v,t}$ is observed, and $M_{v,t} = 0$ if $X_{v,t}$ is missing. The complementary mask $\overline{\mathbf{M}} \in \{0, 1\}^{V \times T}$ is defined by $\overline{M}_{v,t} = 1 - M_{v,t}, \forall v, t$. With a slight abuse of notation, we also regard $\mathbf{M}$ and $\overline{\mathbf{M}}$ as the index sets of the observed and missing entries, so that the observed values are $\mathbf{X}_{\mathbf{M}} = \{X_{v,t} \mid (v, t) \in \mathbf{M}\}$ and the missing values are $\mathbf{X}_{\overline{\mathbf{M}}} = \{X_{v,t} \mid (v, t) \in \overline{\mathbf{M}}\}$. We consider the case where it is known which entries are observed, i.e., $\mathbf{M}$ is known. Thus an incomplete data sample is given by $(\mathbf{X}_{\mathbf{M}}, \mathbf{M})$. An incomplete dataset consists of N such samples, denoted by $\mathcal{D} = \{(\mathbf{X}^{(i)}_{\mathbf{M}^{(i)}}, \mathbf{M}^{(i)})\}_{i=1,2,\ldots,N}$.

Following Reference [35], we model the generative process of incomplete data as follows. A complete data sample $\mathbf{X}$ is first drawn from the complete data distribution $p(\mathbf{X})$. Given $\mathbf{X}$, a mask sample $\mathbf{M}$ is then drawn from the conditional mask distribution $p(\mathbf{M} \mid \mathbf{X})$. The resulted incomplete data sample $(\mathbf{X}_{\mathbf{M}}, \mathbf{M})$ follows the distribution

$$p(\mathbf{X}_{\mathbf{M}}, \mathbf{M}) = \int p(\mathbf{X}) p(\mathbf{M} \mid \mathbf{X}) d\mathbf{X}_{\overline{\mathbf{M}}}.$$

We focus on the **Missing Completely At Random (MCAR)** case [35], where the mask $\mathbf{M}$ is independent of the underlying complete data $\mathbf{X}$, i.e., $p(\mathbf{M} \mid \mathbf{X}) = p(\mathbf{M})$. Our proposed framework

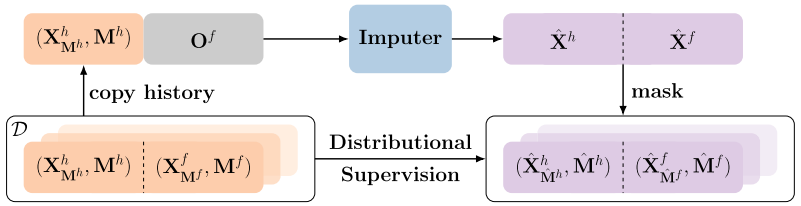


Fig. 1. Supervision.

can be easily generalized to the **Missing At Random (MAR)** case [35] where $\mathbf{M}$ only depends on the observed data $\mathbf{X}_{\mathbf{M}}$, i.e., $p(\mathbf{M} \mid \mathbf{X}) = p(\mathbf{M} \mid \mathbf{X}_{\mathbf{M}})$.

3.2 Problem Statement

Given an incomplete history $(\mathbf{X}^h_{\mathbf{M}^h}, \mathbf{M}^h)$ over T^h timestamps, our task is to predict the future $\mathbf{X}^f$ over the next T^f timestamps. Specifically, we seek a prediction function g_{pred} that takes $(\mathbf{X}^h_{\mathbf{M}^h}, \mathbf{M}^h) \sim p(\mathbf{X}^h_{\mathbf{M}^h}, \mathbf{M}^h)$ as input and outputs the predicted future $\hat{\mathbf{X}}^f$.

We allow g_{pred} to be random to accommodate multiple prediction [35], where multiple predicted samples are provided to reflect the uncertainty. We would like the predicted samples to follow the conditional distribution of the future given the incomplete history,

$$\hat{\mathbf{X}}^f = g_{\text{pred}}\left(\mathbf{X}^h_{\mathbf{M}^h}, \mathbf{M}^h\right) \sim p\left(\mathbf{X}^f \mid \mathbf{X}^h_{\mathbf{M}^h}, \mathbf{M}^h\right).$$

When a single prediction is desired, we can use a summary statistic such as the mean of multiple predicted samples. Note g_{pred} may depend on the underlying graph $\mathcal{G}$, which is part of the input.

The predictor g_{pred} will be trained on an incomplete dataset with missing values in both history and future. A sample in the dataset takes the form of $(\mathbf{X}^h_{\mathbf{M}^h}, \mathbf{M}^h; \mathbf{X}^f_{\mathbf{M}^f}, \mathbf{M}^f)$, where $(\mathbf{X}^f_{\mathbf{M}^f}, \mathbf{M}^f)$ is the incomplete future.

4 METHODOLOGY

We first introduce the proposed ImpGAN framework in Section 4.1. Then we present in Section 4.2 the detailed design of modules that specializes ImpGAN to NETS-ImpGAN.

4.1 The ImpGAN Framework

A key challenge for training g_{pred} is how to use the incomplete future $(\mathbf{X}^f_{\mathbf{M}^f}, \mathbf{M}^f)$ to supervise the complete prediction $\hat{\mathbf{X}}^f$. Supervising only on the observed future values can lead to inferior performance. We overcome this issue by supervising on the joint distribution of incomplete history and future. Specifically, we regard the prediction problem as a special type of imputation problem by treating the future as missing data. In Figure 1, we feed incomplete history $(\mathbf{X}^h_{\mathbf{M}^h}, \mathbf{M}^h)$ into the imputer and obtain the completed history and predicted future $(\hat{\mathbf{X}}^h, \hat{\mathbf{X}}^f)$. Then we mask $(\hat{\mathbf{X}}^h, \hat{\mathbf{X}}^f)$ by a generated mask to obtain an incomplete sample $(\hat{\mathbf{X}}^h_{\hat{\mathbf{M}}^h}, \hat{\mathbf{M}}^h; \hat{\mathbf{X}}^f_{\hat{\mathbf{M}}^f}, \hat{\mathbf{M}}^f)$, the distribution of which is supervised by that of the real data $(\mathbf{X}^h_{\mathbf{M}^h}, \mathbf{M}^h; \mathbf{X}^f_{\mathbf{M}^f}, \mathbf{M}^f)$.

To simplify the notation, we will use $(\mathbf{X}_{\mathbf{M}}, \mathbf{M})$ to denote a full sample of length $T = T^h + T^f$, consisting of both history and future, where $\mathbf{X} = \mathbf{X}^h \| \mathbf{X}^f$, $\mathbf{M} = \mathbf{M}^h \| \mathbf{M}^f$, with $\|$ denoting concatenation in the temporal dimension. By introducing a new mask $\mathbf{M}^* = \mathbf{M}^h \| \mathbf{O}^f$, where $\mathbf{O}^f$ is an all-zero mask of size $V \times T^f$, we can rewrite the input to the imputer as $(\mathbf{X}_{\mathbf{M}^*}, \mathbf{M}^*)$. The output of the imputer is denoted by $\hat{\mathbf{X}} = \hat{\mathbf{X}}^h \| \hat{\mathbf{X}}^f$. The imputed values $\hat{\mathbf{X}}_{\overline{\mathbf{M}^*}}$ consist of both the imputed

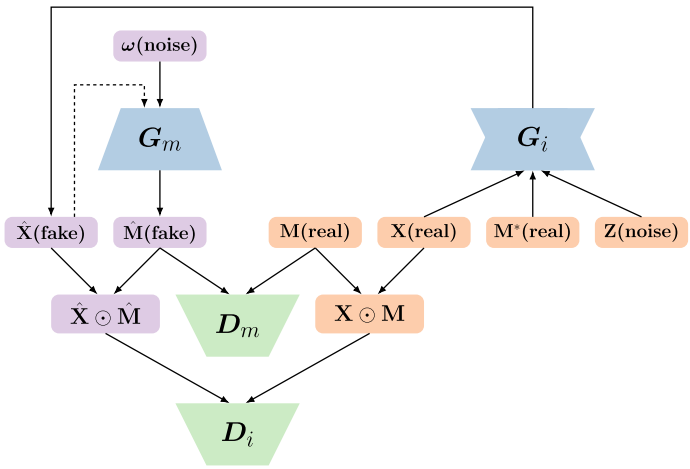


Fig. 2. Architecture of ImpGAN.

history $\hat{\mathbf{X}}^h_{\overline{\mathbf{M}}^h}$ and the predicted future $\hat{\mathbf{X}}^f$. Thus instead of learning g_{pred}, we can learn a random function g_{imp} such that

$$\hat{\mathbf{X}}_{\overline{\mathbf{M}^*}} = g_{\text{imp}}(\mathbf{X}_{\mathbf{M}^*}, \mathbf{M}^*) \sim p(\mathbf{X}_{\overline{\mathbf{M}^*}} \mid \mathbf{X}_{\mathbf{M}^*}, \mathbf{M}^*)$$

and then extract $\hat{\mathbf{X}}^f$. Using the reparameterization trick, we can learn a deterministic function, still denoted by g_{imp}, that takes some random noise $\mathbf{Z}$ as an additional input such that

$$\hat{\mathbf{X}}_{\overline{\mathbf{M}^*}} = g_{\text{imp}}(\mathbf{X}_{\mathbf{M}^*}, \mathbf{M}^*, \mathbf{Z}) \sim p(\mathbf{X}_{\overline{\mathbf{M}^*}} \mid \mathbf{X}_{\mathbf{M}^*}, \mathbf{M}^*).$$

We adopt GAN [19] to learn g_{imp} for its high sampling efficiency and propose a deep learning framework named ImpGAN, as shown in Figure 2. The function g_{imp} is realized by the imputation generator $G_i(\mathbf{X}, \mathbf{M}^*, \mathbf{Z})$, where $\mathbf{M}^* = \mathbf{M}^h \| \mathbf{O}^f$ is the mask indicating that the entire future is missing. The construction of G_i is

$$G_i(\mathbf{X}, \mathbf{M}^*, \mathbf{Z}) = \mathbf{X} \odot \mathbf{M}^* + \hat{G}_i(\mathbf{X} \odot \mathbf{M}^* + \mathbf{Z} \odot \overline{\mathbf{M}^*}) \odot \overline{\mathbf{M}^*},$$

where $\mathbf{Z} \sim p_{\mathbf{Z}}$ is a random noise of size $V \times T$, $\odot$ is element-wise multiplication, and $\hat{G}_i$ is a function whose input and output are both of size $V \times T$. In this article, $\hat{G}_i$ is implemented by the neural network to be introduced in Section 4.2. Note that G_i depends on $\mathbf{X}$ only through the masked form $\mathbf{X} \odot \mathbf{M}^*$, so the true input to G_i is actually $\mathbf{X}_{\mathbf{M}^*}$, and the missing data $\mathbf{X}_{\overline{\mathbf{M}^*}}$ is never needed. The masking by $\mathbf{M}^*$ outside of $\hat{G}_i$ is used to retain the observed data. If G_i successfully captures the conditional distribution $p(\mathbf{X}_{\overline{\mathbf{M}^*}} \mid \mathbf{X}_{\mathbf{M}^*}, \mathbf{M}^*)$, then the output $\hat{\mathbf{X}}$ will have the distribution $p(\mathbf{X})$ of $\mathbf{X}$.

To train G_i, we need to use the incomplete data $(\mathbf{X}_{\mathbf{M}}, \mathbf{M})$ in $\mathcal{D}$ to supervise the output $\hat{\mathbf{X}}$. This is done by re-masking $\hat{\mathbf{X}}$ back to an incomplete sample, which we then have the imputation discriminator D_i try to distinguish from the real incomplete data. More specifically, we use a standard GAN $(G_m(\boldsymbol{\omega}), D_m(\mathbf{M}))$ to learn the mask distribution $p(\mathbf{M})$, as the mask $\mathbf{M} = \mathbf{M}^h \| \mathbf{M}^f$ is fully observed. We then generate a mask $\hat{\mathbf{M}}$ using G_m and obtain the corresponding incomplete sample $(\hat{\mathbf{X}}_{\hat{\mathbf{M}}}, \hat{\mathbf{M}})$, which D_i tries to distinguish from real samples $(\mathbf{X}_{\mathbf{M}}, \mathbf{M})$ in $\mathcal{D}$. Since neural networks generally take arrays of fixed shape as input, we fill missing entries with zeros and have D_i distinguish between $\mathbf{X} \odot \mathbf{M}$ and $\hat{\mathbf{X}} \odot \hat{\mathbf{M}}$ instead. Note that the input $\mathbf{X} \odot \mathbf{M} = (\mathbf{X}^h_{\mathbf{M}^h} \odot \mathbf{M}^h) \| (\mathbf{X}^f_{\mathbf{M}^f} \odot \mathbf{M}^f)$ into D_i includes partially observed future, while the input $\mathbf{X} \odot \mathbf{M}^* = (\mathbf{X}^h_{\mathbf{M}^h} \odot \mathbf{M}^*) \| \mathbf{O}^f$ into G_i includes no future values. In this way, we are able to use only incomplete future to supervise prediction.

An alternative way of training G_i is provided in MisGAN [31], which learns the complete data distribution $p(\mathbf{X})$ from $(\mathbf{X_M}, \mathbf{M})$ and uses the learned distribution $\hat{p}(\mathbf{X})$ to supervise imputation. However, as we will be see in Section 5.4.2, regarding the imperfectly learned $\hat{p}(\mathbf{X})$ as ground truth will result in error accumulation, which ImpGAN avoids by directly using incomplete data for supervision. The imperfectly learned mask distribution $\hat{p}(\mathbf{M})$ could also cause error accumulation, but we will see in Section 5.4.2 that this effect is negligible, as $p(\mathbf{M})$ is easy to learn.

Following the Wasserstein GAN [5] formulation, we define the training objectives by

$$\min_{G_m} \max_{D_m \in \mathcal{F}_m} \mathcal{L}_m(D_m, G_m), \quad \min_{G_i} \max_{D_i \in \mathcal{F}_i} \mathcal{L}_i(D_i, G_i, G_m),$$

where $\mathcal{F}_m$ and $\mathcal{F}_i$ are the classes of 1-Lipschitz functions for D_m and D_i, respectively. The loss functions are

$$\begin{aligned}\mathcal{L}_m(D_m, G_m) &= \mathbb{E}[D_m(\mathbf{M})] - \mathbb{E}[D_m(G_m(\boldsymbol{\omega}))],\\ \mathcal{L}_i(D_i, G_i, G_m) &= \mathbb{E}[D_i(\mathbf{X} \odot \mathbf{M})] - \mathbb{E}[D_i(G_i(\mathbf{X}, \mathbf{M}^*, \mathbf{Z}) \odot G_m(\boldsymbol{\omega}))]\\ &\quad + \beta \mathbb{E}\|\hat{G}_i(\mathbf{X}, \mathbf{M}^*, \mathbf{Z}) \odot \mathbf{M}^* - \mathbf{X} \odot \mathbf{M}^*\|_1,\end{aligned}$$

where the expectations are taken over $(\mathbf{X}, \mathbf{M}) \sim p_{\mathcal{D}}$, $\boldsymbol{\omega} \sim p_{\omega}$, and $\mathbf{Z} \sim p_{\mathbf{Z}}$, $p_{\mathcal{D}}$ is the underlying distribution of $\mathcal{D}$, $\|\cdot\|_1$ is the L_1-norm, and β is the tradeoff parameter for L_1-norm. The L_1-norm term is the reconstruction loss for the output of $\hat{G}_i$, which is added to force the output of $\hat{G}_i$ to have the same observed history as the input.

Since we focus on the MCAR case, the mask generator $G_m(\boldsymbol{\omega})$ only takes random noise $\boldsymbol{\omega}$ as input. To generalize to the MAR case, G_m can take $\hat{\mathbf{X}}$ as additional input to model the dependence of mask $\mathbf{M}$ on the underlying $\mathbf{X}$, as indicated by the dotted line in Figure 2.

ImpGAN can also be turned into a generic imputation framework of independent interest. Recall that the data $\mathbf{X} \odot \mathbf{M}^*$ fed into G_i include no future values and differ from the data $\mathbf{X} \odot \mathbf{M}$ fed into D_i. This is a consequence of the prediction task. If we feed the same data $\mathbf{X} \odot \mathbf{M}$ into both G_i and D_i, then ImpGAN will become a generic imputation framework that can be combined with different designs for the generators and discriminators. For the prediction task, the discrepancy between $\mathbf{X} \odot \mathbf{M}^*$ and $\mathbf{X} \odot \mathbf{M}$ requires G_i to be able to capture temporal correlations at the minimum.

4.2 Graph Temporal Attention Networks

4.2.1 GTA U-Net. We specialize ImpGAN to NETS-ImpGAN by designing GTANs for the generators and discriminators that capture both inter-time-series and temporal correlations of incomplete NETS. This section presents the **Graph Temporal Attention U-Net (GTA U-Net)** that we propose for the imputation generator G_i. The mask generator and the discriminators have similar building blocks; see Section 4.2.2 for their detailed architectures.

Figure 3 shows the architecture of GTA U-Net. We use **Graph Attention Network (GAT)** [53] to capture the inter-time-series correlations, and we use Multi-Head Self-Attention [52] and **Temporal Convolution (T-Conv)** to capture global and local temporal correlations, respectively. The input is $\mathbf{Y} = \mathbf{X} \odot \mathbf{M}^* + \mathbf{Z} \odot \overline{\mathbf{M}^*}$, consisting of both observed values and random noise. As noted in Reference [66], this may lead to inferior performance due to error accumulation. The incorporated attention mechanisms can help mitigate error accumulation by differentiating the observed and filled values.

Similarly to U-Net [45], GTA U-Net has a U-shaped structure, where the contractive encoding path (left side) first extracts a lower-dimensional representation of the input incomplete data and the expansive decoding path (right side) then recovers the complete data from this representation. Along the encoding path, the input first goes through a GAT layer, then a Multi-Head

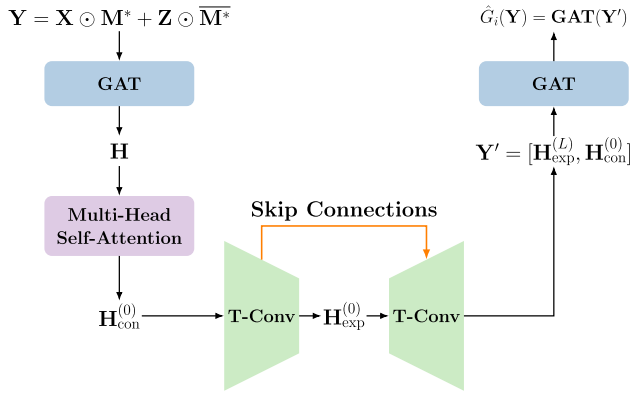


Fig. 3. Architecture of GTA U-Net.

Self-Attention layer, and, finally, a stack of T-Conv layers. Along the decoding path, data goes through a stack of T-Conv layers and then a GAT layer.

Graph Attention Layer. The GAT layer on the contractive path takes $\mathbf{Y}$ as input and outputs $\mathbf{H} = (H_{i,t}) = \text{GAT}(\mathbf{Y}) \in \mathbb{R}^{V \times T}$ given by

$$H_{i,t} = \sigma\left(\alpha_{i,i,t}\theta Y_{i,t} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j,t}\theta Y_{j,t}\right), \quad i \in \mathcal{V}, t \in \mathcal{T}, \tag{1}$$

where $\mathcal{N}(i)$ is the set of one-hop neighbors of node i, $\theta \in \mathbb{R}$ is a training parameter, and σ is an activation function. The attention coefficient $\alpha_{i,j,t}$ is given by

$$\alpha_{i,j,t} = \frac{\exp\left(\text{LeakyReLU}(\mathbf{a}^\top[\theta Y_{i,t} \| \theta Y_{j,t}])\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\text{LeakyReLU}(\mathbf{a}^\top[\theta Y_{i,t} \| \theta Y_{k,t}])\right)}, \tag{2}$$

where $\|$ is concatenation and $\mathbf{a} \in \mathbb{R}^2$ is a training parameter.

Multi-Head Self-Attention Layer. Multi-Head Self-Attention captures the temporal correlations from a global view of all the timestamps. The input to the Multi-Head Self-Attention layer is $\mathbf{H}$. Multi-Head Self-Attention consists of multiple heads. For the ith head, define query $\mathbf{Q}_i = \mathbf{H}^\top \mathbf{W}_i^Q \in \mathbb{R}^{T \times d_k}$, key $\mathbf{K}_i = \mathbf{H}^\top \mathbf{W}_i^K \in \mathbb{R}^{T \times d_k}$, and value $\mathbf{V}_i = \mathbf{H}^\top \mathbf{W}_i^V \in \mathbb{R}^{T \times d_v}$, where $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V$ are training parameters. The output of the ith head is

$$\text{head}_i = \text{Softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d_k}}\right)\mathbf{V}_i.$$

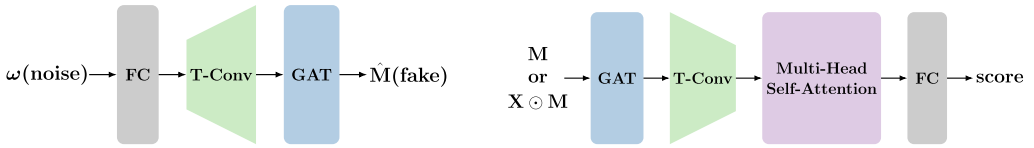
The output of Multi-Head Self-Attention is then

$$\text{MultiHead} = \mathbf{W}^O \text{Concat}(\text{head}_1, \ldots, \text{head}_h)^\top,$$

where $\text{Concat}(\cdot)$ is concatenation along the second dimension and $\mathbf{W}^O \in \mathbb{R}^{V \times hd_v}$ is a training parameter.

Temporal Convolution Layers. T-Conv captures the temporal correlations from a local view of the neighboring timestamps, which complements Multi-Head Self-Attention. The convolutional structure also avoids error accumulation, which recurrent structures potentially suffer from. This leads to higher stability in multi-step prediction, as will be demonstrated in Section 5.2.

We use 1D CNN [54] to halve and double the temporal dimensions along the contractive path and the expansive path, respectively. There are L layers along each path. Let $\mathbf{H}^{(0)}_{\text{con}} \in \mathbb{R}^{V \times T}$ denote



(a) Mask Generator (b) Discriminators

Fig. 4. Architecture of mask generator and the discriminators.

the output of Multi-Head Self-Attention and $\mathbf{H}_{\text{con}}^{(l)} \in \mathbb{R}^{V \times T/2^l}$ that of the lth T-Conv layer along the contractive path. For $l = 1, \ldots, L$,

$$\mathbf{H}_{\text{con}}^{(l)} = \sigma\left(\text{BN}\left(\text{Conv1d}\left(\mathbf{H}_{\text{con}}^{(l-1)}, \text{stride} = 2\right)\right)\right),$$

where Conv1d$(\cdot, \text{stride} = 2)$ is a 1D CNN with stride 2, and BN$(\cdot)$ is batch normalization.

The output $\mathbf{H}_{\text{con}}^{(L)}$ of the contractive path is sent to the expansive path. Let $\mathbf{H}_{\text{exp}}^{(0)} = \mathbf{H}_{\text{con}}^{(L)}$ and denote by $\mathbf{H}_{\text{exp}}^{(l)} \in \mathbb{R}^{V \times T/2^{L-l}}$ the output of the lth T-Conv layer along the expansive path. For $l = 1, \ldots, L$,

$$\mathbf{H}_{\text{exp}}^{(l)} = \sigma\left(\text{BN}\left(\text{DeConv1d}\left(\left[\mathbf{H}_{\text{exp}}^{(l-1)}, \mathbf{H}_{\text{con}}^{(L-l+1)}\right], \text{stride} = 2\right)\right)\right),$$

where DeConv1d$(\cdot, \text{stride} = 2)$ is a 1D de-convolution with stride 2 and $[\cdot, \cdot]$ denotes concatenation along the feature dimension.

We have also followed U-Net [45] to add skip connections as indicated by the orange arrow in Figure 3. They are used to preserve information at the border, i.e., at the starting and ending timestamps, since such information tends to be lost in convolution.

The expansive path is followed by a final GAT layer, which takes as input $\mathbf{Y}' = [\mathbf{H}_{\text{exp}}^{(L)}, \mathbf{H}_{\text{con}}^{(0)}] \in \mathbb{R}^{V \times T \times 2}$, and outputs

$$\hat{G}_i(\mathbf{Y}) = \text{GAT}(\mathbf{Y}').$$

The GAT layer has a similar structure as in Equations (1) and (2), except that the training parameter $\theta \in \mathbb{R}$ is replaced by $\boldsymbol{\Theta} \in \mathbb{R}^{1 \times 2}$, since $\mathbf{Y}'$ has 2D feature, but the output GAT$(\mathbf{Y}') \in \mathbb{R}^{V \times T}$ has only 1D feature.

4.2.2 Mask Generator and the Discriminators. Mask generator G_m, mask discriminator D_m and imputation discriminator D_i have similar building blocks to those of GTA U-Net.

The architecture of G_m is shown in Figure 4(a). The random noise $\boldsymbol{\omega}$ first goes through a **Fully Connected (FC)** layer, which reshapes $\boldsymbol{\omega}$ to the desired shape. Then the reshaped data goes through L layers of T-Conv and a GAT layer, which is the same as the expansive path of GTA U-Net without skip connections.

The discriminators D_m and D_i have the same architecture shown in Figure 4(b). The input goes through a GAT layer, a Multi-Head Self-Attention layer, and L layers of T-Conv, which is the same as the contractive path of GTA U-Net but in the reverse order. This is followed by a FC layer, whose output is a scalar score that indicates how real the input is.

5 EVALUATION

We introduce the experimental setup in Section 5.1. Section 5.2 presents the prediction performance of NETS-ImpGAN. Section 5.3 gives the comparison with two-phase methods that first impute incomplete data and then use the imputed data to train prediction models. Section 5.4 gives the efficacy study. Section 5.5 gives the efficiency study.

5.1 Experimental Setup

5.1.1 Datasets. We evaluate NETS-ImpGAN on the following four real-world datasets.

Metro Passenger Flow (Metro) [50] is a collection of passenger outbound flows from 81 metro stations in Hangzhou, China, within every 10 minutes in January 2019. This dataset is collected from automated fare collection records. In the underlying graph, nodes represent the metro stations, and edges are constructed based on pairwise transition probability between stations. In the presence of missing data, we first delete the automated fare collection records that correspond to the missing part of all the incomplete samples and then follow Ou et al. [44] to compute the transition probability matrix with the remaining records. A pair of nodes are connected by an undirected edge if at least one of the transition probabilities between them is no less than 0.02.

Air Quality Index (Air) [42] is a collection of air quality records from 437 monitoring stations in China within every hour from May 2014 to April 2015. Same as Cini et al. [12], we focus on the PM2.5 pollutant. In the underlying graph, nodes represent the monitoring stations, and edges are constructed based on pairwise geographic distance between stations. We follow Cini et al. [12] that processes the distance with threshold Gaussian kernel. A pair of nodes are connected by an undirected edge if the processed distance between them is no less than 0.1.

Electricity Consumption (Electricity) [4] is a collection of electricity consumption records from 485 smart meters in Ireland within every 30 minutes from 2009 to 2010. In the underlying graph, nodes represent the smart meters, and edges are constructed based on the similarity between smart meters. We follow Cini et al. [12] that first computes a similarity matrix under correntropy and then builds a k-nearest-neighbor graph where $k = 10$.

All the above three datasets consist of complete samples. We generate incomplete datasets by introducing missing values according to the missing patterns in Section 5.1.2.

Traffic Speed (Speed) [14] is a collection of traffic speed records from 1,343 road segments in Chengdu, China, within every hour in 2018. In the underlying graph, nodes represent the road segments, and edges are constructed based on the adjacency of road segments. This dataset is naturally incomplete with an average missing rate of 30%.

5.1.2 Missing Patterns. We follow previous works [10, 31, 32, 36–38, 40, 49, 61] to evaluate the Random pattern given below and also consider the more general **Multiple Block of Variable Shape (MV)** pattern. The missing rate, which is denoted by r, is set to be low, medium, and high at 25%, 50%, and 75%, respectively, and also very low at 2%, 4%, 6%, 8%, and 10%.

- *Random.* In each sample, values on all nodes at all timestamps are independently randomly missing with probability r.
- *MV.* In each sample, multiple blocks are missing. In each block, values on N_v nodes at randomly selected N_t consecutive timestamps are missing, where N_v and N_t are uniformly randomly drawn from $[l_v, u_v]$ and $[l_t, u_t]$. The nodes are selected by breadth-first search from a random node until the desired number of nodes have been traversed. The number of blocks is $\lfloor \frac{VTr}{(l_v+u_v)(l_t+u_t)/4} \rfloor$. In this article, we set $l_v = 1$, $u_v = 7$, $l_t = 1$, and $u_t = 3$. The blocks are selected independently, so the missing rate may not be exactly r due to possible overlapping of the blocks.

5.1.3 Baselines. We compare NETS-ImpGAN with the following prediction baselines: (1) simple methods including Mean, **Temporal Linear Extrapolation (TLE)**, and **Last Observation (LO)**, and (2) state-of-the-art methods, including DCMF [8], TRMF [63], BRITS [10], E^{2}GAN [36, 37], GRIN [12], CSDI [49], and mSSA [1]. For a fair comparison, we focus on methods that do not require complete data for training. Those that require complete data for training will be compared in Section 5.3. Mean, TLE, and LO are described as follows.

Table 1. Differences between NETS-ImpGAN and Prediction Baselines

	Supervision with Incomplete Future	Graph	Temporal
Mean	×	×	×
TLE	×	×	√
LO	×	×	√
DCMF	×	√	√
TRMF	×	×	√
BRITS	×	×	√
E^2GAN	√	×	√
GRIN	√	√	√
CSDI	×	×	√
mSSA	×	×	√
NETS-ImpGAN	√	√	√

— *Mean.* Mean predicts the missing entries with the mean of observed entries in the same sample.
— *TLE.* TLE predicts a missing entry with the linear function that passes through its last two observed historical entries on the same node in the same sample. Mean is used if the required entries are missing.
— *LO.* LO predicts a missing entry with the last observed historical entry on the same node in the same sample. Mean is used if the required entries are missing.

For DCMF, BRITS, E^2GAN, GRIN, and CSDI, we use them for prediction by treating the future as missing data. For all baselines, we will report the better of the results refined by MIRACLE [30] or not. We will report our performance both with and without the refinement. Note that Facets [9] and NetDyna [21] are reduced to DCMF in our problem, so we only select DCMF as baseline.

Table 1 summarizes the differences between NETS-ImpGAN and the prediction baselines w.r.t. whether they can (1) supervise with incomplete future, (2) exploit the underlying graph, and (3) capture temporal correlations.

5.1.4 Evaluation Metrics. We use **Mean Absolute Error (MAE)**, **Root Mean Square Error (RMSE)**, and **Mean Absolute Percentage Error (MAPE)** as evaluation metrics. Since we learn a distribution of the future, we have the flexibility to optimize the prediction results according to a given metric. Specifically, for MAE and RMSE, we use the sample medium and mean, respectively, of multiple samples generated by the imputation generator; for MAPE, we solve an empirical MAPE minimization problem by quantile regression [27]. Since DCMF, E^2GAN, and CSDI also have distributions, we do the same optimization for them. The metrics are computed over all T^f future timestamps unless stated otherwise. We will mostly report the results for MAE due to space limit, but the other results are similar.

5.1.5 Implementation Details. Each generated sample has $T = 16$ timestamps; the first 8 constitute history, and the last 8 future. Since the ranges of values vary greatly across nodes, we apply Min-Max normalization to scale the data to $[-1, 1]$ for each node before they are fed into the neural networks. The predicted or imputed values are then rescaled back to the original range.

The proposed NETS-ImpGAN is implemented with PyTorch. All Multi-Head Self-Attention modules have $h = 3$ heads. The contractive and expansive paths have $L = 3$ T-Conv layers, respectively. All T-Conv modules have kernel of size 3×1 and stride of 2, and the number of channels doubles along the contractive path and halves along the expansive path. The noise $\boldsymbol{\omega}$ fed into the mask generator G_m is a 128-dimensional standard Gaussian random vector. For activation functions, we

Table 2. Prediction Performance on Metro, Air, and Electricity under 25% Missing Rate

(a) Random

Method	Metro			Air			Electricity		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Mean	104.63	169.79	164.75	45.24	61.20	37.56	2.98	4.64	86.63
TLE	148.35	195.67	231.53	50.42	86.23	45.32	4.10	5.50	121.42
LO	54.54	106.83	85.24	26.42	40.11	20.99	1.65	3.01	43.25
DCMF	45.81	86.30	64.97	16.84	31.29	14.33	1.42	2.32	33.63
TRMF	53.64	101.38	87.86	24.41	38.13	18.39	1.53	2.97	45.14
BRITS	46.39	91.89	78.92	20.92	33.20	15.03	1.39	2.77	41.69
E^2GAN	80.25	139.48	124.97	31.75	44.58	26.05	2.22	3.78	65.94
GRIN	*38.53	*76.16	*49.94	*16.28	*29.60	*11.70	*1.26	*2.15	*25.93
CSDI	44.78	93.48	59.29	18.39	33.94	14.98	1.36	2.42	38.94
mSSA	50.39	96.93	84.29	22.39	35.59	17.10	1.49	2.99	44.88
NETS-ImpGAN	33.23	66.43	32.25	15.78	26.83	10.04	1.17	1.98	17.32
NETS-ImpGAN†	**30.38**	**64.18**	**31.14**	**13.27**	**24.45**	**9.29**	**1.06**	**1.65**	**15.23**

(b) MV

Method	Metro			Air			Electricity		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Mean	101.36	156.39	158.49	44.23	56.23	37.24	2.81	4.25	81.58
TLE	154.63	191.32	237.24	52.05	85.38	46.42	4.28	5.24	126.03
LO	53.10	101.37	79.25	25.29	40.13	20.88	1.59	2.89	41.90
DCMF	44.36	81.11	45.24	16.30	30.42	11.25	1.39	2.26	24.77
TRMF	52.35	109.26	86.62	24.85	40.90	19.94	1.61	2.99	45.57
BRITS	48.25	98.17	78.30	20.63	38.75	15.15	1.49	2.56	42.53
E^2GAN	72.89	140.25	113.58	28.05	45.28	23.26	2.09	3.72	56.83
GRIN	*44.35	*80.97	*44.78	*15.83	*30.17	*11.05	*1.33	*2.18	*23.90
CSDI	46.93	92.36	57.11	18.29	36.39	14.39	1.50	2.67	43.92
mSSA	51.03	103.59	80.93	23.49	39.61	18.76	1.59	2.97	44.24
NETS-ImpGAN	38.71	62.46	33.90	14.46	28.36	10.55	1.31	2.05	18.21
NETS-ImpGAN†	**35.36**	**59.09**	**32.74**	**13.04**	**25.99**	**9.69**	**1.20**	**1.77**	**15.74**

The results of the baselines are the better of those refined by MIRACLE or not, and the results of NETS-ImpGAN† are refined by MIRACLE. **Bold**, underline, and superscript * indicate the best, second best, and third best in each column. These marks will also be used in Table 5.

use `LeakyReLU` throughout, except for the final GAT layer of G_i and G_m, where `tanh` and `hardtanh` are used, respectively, to make the range of the generated data and mask consistent with that of the normalized real data and real mask.

We use 90% of the samples for training, 5% for validation, and 5% for testing. The model is trained for 1,000 epochs with batch size 64. We follow the common practice that alternatively optimizes the discriminators for 5 epochs and the generators for 1 epoch. We use the Adam optimizer with learning rate 0.0001. The tradeoff parameter β in Section 4.1 is set to 10. In the testing phase, we randomly generate 10 samples by G_i to compute the optimal prediction as specified in Section 5.1.4.

5.2 Prediction Performance

5.2.1 Comparison with Baselines. We compare NETS-ImpGAN with the baselines under different missing patterns and missing rates. Table 2 shows the results on Metro, Air, and Electricity

Table 3. Prediction Performance on Speed, Whose Missing Rate Is 30%

	Mean	TLE	LO	DCMF	TRMF	BRITS	E^{2}GAN	GRIN	CSDI	mSSA	NETS-ImpGAN	NETS-ImpGAN†
MAE	42.69	117.49	37.53	20.35	30.88	24.43	35.92	*16.69	21.70	27.49	13.45	**12.16**
RMSE	43.94	125.49	36.87	23.17	31.72	25.39	36.93	*18.75	22.48	28.70	15.28	**13.10**
MAPE	97.25	196.28	74.82	41.48	62.67	48.16	73.85	*37.74	43.36	53.84	32.68	**30.32**

The results of the baselines are the better of those refined by MIRACLE or not, and the results of NETS-ImpGAN† are refined by MIRACLE. **Bold**, underline, and superscript * indicate the best, second best, and third best in each row.

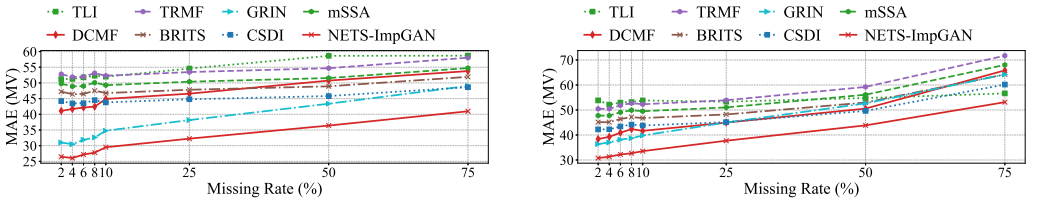


Fig. 5. Prediction performance with increasing missing rate on Metro. The results of the baselines are the better of those refined by MIRACLE or not, and the results of NETS-ImpGAN is without refinement.

under 25% missing rate; those under the other settings are similar and omitted due to space limit. NETS-ImpGAN outperforms the baselines even when they are refined by MIRACLE, reducing the MAE of the second best GRIN by 2%–16%, with an average of 9%, reducing the RMSE of GRIN by 6%-22%, with an average of 11%, and reducing the MAPE of GRIN by 4%–34%, with an average of 20%. Table 3 shows the results on the naturally incomplete Speed dataset, whose missing rate is 30%. The metrics are only computed on the observed future entries, since the ground truth of missing future is unavailable. NETS-ImpGAN outperforms the refined GRIN, reducing MAE by 21%, RMSE by 17%, and MAPE by 15%, which shows that NETS-ImpGAN can be applied to data with complex real-world missingness. When refined by MIRACLE, NETS-ImpGAN further reduces the MAE by 10%, the RMSE by 8%, and MAPE by 6% averaged over all four datasets. MIRACLE has similar level of improvement on the baselines and NETS-ImpGAN.

Note that Mean, TLE, and LO have strong implicit assumptions on the data distribution. Mean assumes low temporal and inter-time-series variation, TLE assumes linearity in the temporal dimension, and LO assumes low temporal variation. Since these assumptions hardly hold, these models have poor performance in general. In contrast, TRMF, BRITS, E^{2}GAN, CSDI, and mSSA can capture temporal correlations without such restrictive assumptions and thus achieve better performance than Mean and TLE, but they do not exploit the underlying graph. DCMF and GRIN can further exploit the graph, leading to even better performance. However, DCMF relies on the assumed linear system model, GRIN has no direct control on the missing part of future, and its bidirectional architecture may not be a perfect match for prediction, which results in their worse performance than NETS-ImpGAN.

5.2.2 Performance with Increasing Missing Rate. Figure 5 shows the performance against missing rate on Metro. Since Mean, TLE, and E^{2}GAN have much poorer performance than the others, we omit them in the figure for better visualization and easy comparison. Note that NETS-ImpGAN consistently outperforms the baselines across different missing rates and its performance degrades gracefully as missing rate increases. The results on Air and Electricity are similar.

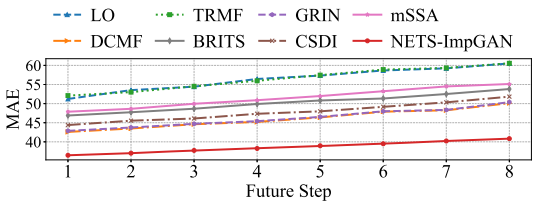


Fig. 6. Stepwise prediction performance on Metro under MV and 25% missing rate. The results of the baselines are the better of those refined by MIRACLE or not, and the results of NETS-ImpGAN is without refinement.

5.2.3 Stepwise Performance. In this section, we study the prediction performance for individual future timesteps rather than the overall performance. Figure 6 shows the MAE on Metro under MV and 25% missing rate against the number of steps into the future; those under the other settings are similar. We also omit Mean, TLE, and E^{2}GAN. Note that NETS-ImpGAN has the best performance acoss all timesteps. In particular, it performs much better than the baselines in long-term prediction. Its accuracy for the eighth step is higher than those of the others even for the first step. Moreover, its performance degrades more slowly than the others as timestep increases. This is partly because the convolution structure in GTAN helps avoid error accumulation typically suffered by recurrent structures in BRITS and DCMF.

5.3 Comparison with Two-phase Methods

A common alternative way to deal with missing data is to use two-phase methods, which first impute incomplete data and then use the imputed data to train prediction models. In this section, we demonstrate the advantage of NETS-ImpGAN over two-phase methods. To make things clear, in the rest of this section, we will distinguish the two usages of NETS-ImpGAN. It is referred to as NETS-ImpGAN when used for imputation, and NETS-ImpGAN* for prediction, where the superscript star alludes to the mask $\mathbf{M}^*$ used for prediction. We first show that NETS-ImpGAN achieves the state-of-the-art imputation performance. We then show that NETS-ImpGAN* outperforms two-phase methods even when NETS-ImpGAN is used for imputation.

5.3.1 Imputation Performance. In this section, we evaluate the performance of NETS-ImpGAN for imputing the incomplete history. The baselines include Mean, DCMF, TRMF, BRITS, E^{2}GAN, GRIN, CSDI, and mSSA, all of which are also used in Section 5.2. We also add five more imputation baselines as follows: (1) simple methods, including **Neighborhood Average (NA)** and **Temporal Linear Interpolation (TLI)**, and (2) state-of-the-art methods including WDGTC [34], S-MKKM [18], and SD-ADMM [41]. NA and TLI are described below.

— *NA.* NA imputes a missing entry with the mean of observed values on its one-hop neighbors in the underlying graph at the same timestamp in the same sample. Mean is used if the required entries are missing.

— *TLI.* TLI imputes a missing entry with the mean of its last-observed historical and first-observed future entries on the same node in the same sample. Mean is used if the required entries are missing.

As we will use multiple imputation in two-phase prediction, in addition to MAE, we also measure the imputation quality using the **Wasserstein Distance (WD)** between the empirical distributions of real complete samples and imputed samples [58].

Table 4 summarizes the differences between NETS-ImpGAN and the additional imputation baselines w.r.t. whether they can (1) supervise with incomplete data, (2) exploit the underlying graph, and (3) capture temporal correlations.

Table 4. Differences between NETS-ImpGAN and Imputation Baselines

	Supervision with Incomplete Data	Graph	Temporal
NA	×	√	×
TLI	×	×	√
WDGTC	×	√	×
S-MKKM	×	√	×
SD-ADMM	×	×	√
NETS-ImpGAN	√	√	√

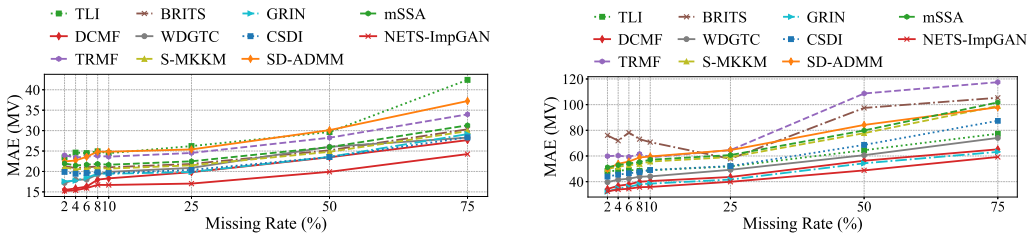


Fig. 7. Imputation performance with increasing missing rate on Metro. The results of the baselines are the better of those refined by MIRACLE or not, and the results of NETS-ImpGAN is without refinement.

Table 5 shows the results of single and multiple imputations on Metro, Air, and Electricity under 25% missing rate, where single imputation is measured by MAE and multiple imputation is measured by WD; those under the other missing rates are similar. Since the error on observed entries is always zero in the imputation task, we do not compare on Speed. NETS-ImpGAN achieves the best performance on all the datasets in both single and multiple imputations. Note that our reported results is different from those in Cini et al. [12] due to different experimental settings. Figure 7 shows the MAE of single imputation against missing rate on Metro; the other settings are similar. We omit Mean, NA, and E^{2}GAN due to their poor performance. Similarly to the prediction case, NETS-ImpGAN outperforms the baselines and the performance degrades gracefully with increasing missing rate. Note that the curves for BRITS bend up at very low missing rates due to overfitting. Comparison with baselines on downstream prediction task will be shown in Section 5.3.2.

5.3.2 Two-phase Methods. We compare NETS-ImpGAN* with two-phase methods that use various combinations of imputation and prediction methods. For the prediction phase in two-phase methods, we use our GTA U-Net and 10 state-of-the-art methods that require complete data for training: those requiring complete input, including DCRNN [33], STGCN [62], Graph WaveNet [57], and LSGCN [23], and those do not, including STGNN-DAE [29], IGNNK [56], SSSDS4 [2], RIHGCN [66], GSTAE [55], and GENIE [15]. RIHGCN can take incomplete history as input but requires complete future for supervision, thus we only impute the future part. Since RIHGCN is designed for NETS with multiple underlying graphs, we adapt its multi-graph module to single graph. Note that we do not select the methods that additionally require periodic or scenario-specific auxiliary information, since there is no such information in our setting even if the data are complete. For the imputation phase, we use all the methods in Section 5.3.1 and also include the commonly used zero imputation. Note that NETS-ImpGAN is used with both single and multiple imputation. The results are shown in Table 6. Due to space limit, we omit those of STGCN, LSGCN, STGNN-DAE, SSSDS4, and RIHGCN that mostly have poorer performance.

Table 5. Imputation Performance on Metro, Air, and Electricity under 25% Missing Rate

(a) Random

Method	Metro		Air		Electricity	
	MAE	WD	MAE	WD	MAE	WD
Mean	148.62	2870.13	65.38	1272.52	4.12	74.94
NA	140.83	2251.52	47.17	769.99	3.95	57.26
TLI	26.72	543.08	12.88	260.95	0.95	13.85
DCMF	19.83	420.94	8.99	176.35	0.85	11.24
TRMF	24.26	545.74	10.65	250.04	0.91	13.20
BRITS	21.52	483.59	9.60	213.85	0.89	12.89
E^2GAN	74.63	1598.37	29.24	613.40	2.08	40.25
WDGTC	20.25	478.24	9.25	208.73	0.89	13.01
S-MKKM	21.50	484.23	9.63	219.20	0.95	14.96
GRIN	*19.57	*413.60	*8.34	*153.57	*0.82	*10.17
CSDI	20.93	479.72	9.18	201.13	0.87	12.65
SD-ADMM	25.39	533.90	11.79	261.31	0.99	14.85
mSSA	22.49	523.59	10.16	223.69	0.88	12.82
NETS-ImpGAN	17.03	289.11	8.22	141.55	0.76	7.76
NETS-ImpGAN†	**16.36**	**257.33**	**8.17**	**133.70**	**0.70**	**6.58**

(b) MV

Method	Metro		Air		Electricity	
	MAE	WD	MAE	WD	MAE	WD
Mean	172.97	2549.77	77.52	1125.75	4.74	62.74
NA	118.42	2185.36	40.25	735.08	3.33	53.95
TLI	27.99	569.15	13.28	246.17	1.19	14.59
DCMF	21.43	399.41	9.06	189.23	0.91	13.57
TRMF	25.74	537.86	11.72	280.69	1.12	15.30
BRITS	23.47	460.58	10.15	222.98	0.96	13.92
E^2GAN	75.32	1632.25	31.61	658.42	2.68	53.77
WDGTC	21.88	503.71	10.40	238.56	1.05	14.70
S-MKKM	22.47	491.44	9.46	213.57	1.09	15.34
GRIN	*21.35	*407.16	*8.65	*163.08	*0.63	*6.93
CSDI	21.94	466.69	9.39	245.86	0.70	7.49
SD-ADMM	26.14	559.03	13.79	301.98	1.15	15.61
mSSA	23.91	548.90	11.78	286.40	0.99	13.82
NETS-ImpGAN	18.17	319.45	8.07	135.81	0.51	6.17
NETS-ImpGAN†	**17.79**	**307.50**	**7.78**	**121.39**	**0.47**	**5.88**

The results of the baselines are the better of those refined by MIRACLE or not, and the results of NETS-ImpGAN† are refined by MIRACLE.

Among the prediction methods, the variants using GTA U-Net, Graph WaveNet, GSTAE, and GENIE have similar performance, slightly better than those of DCRNN and IGNNK; among the imputation methods, the variants using NETS-ImpGAN outperforms the others. Multiple imputation performs slightly better than single imputation, potentially because it better reflects the uncertainty in imputation. However, all the two-phase methods are outperformed by NETS-ImpGAN*.

Table 6. End-to-end vs. two-phase prediction on Metro under 25% missing rate. Each row corresponds to an imputation method, and each column corresponds to a prediction method. The results of the baselines are the better of those refined by MIRACLE or not, and the results of NETS-ImpGAN is without refinement. **Bold** and underline represent the best along each column and row, respectively. Our end-to-end prediction is shown at the bottom.

(a) Random

	DCRNN	Graph WaveNet	IGNNK	GSTAE	GENIE	GTA U-Net
Zero	68.36	67.48	68.56	67.10	67.16	65.32
Mean	67.03	67.68	68.32	66.86	66.51	64.53
NA	66.52	65.02	65.67	63.99	64.25	61.99
TLI	62.45	61.63	62.10	61.20	62.35	60.13
DCMF	49.43	48.20	48.83	47.33	47.90	46.24
TRMF	58.43	59.13	59.35	59.15	59.23	57.25
BRITS	53.67	53.01	53.88	53.17	53.28	52.16
E^2GAN	64.89	64.12	64.57	63.50	63.28	61.89
WDGTC	46.57	45.34	46.32	45.26	45.11	43.39
S-MKKM	54.39	54.92	55.90	54.86	56.23	53.91
GRIN	41.08	41.09	41.53	41.72	41.56	39.16
CSDI	43.48	42.29	44.56	42.55	43.78	41.75
SD-ADMM	58.35	57.60	58.38	57.77	58.95	56.25
mSSA	55.19	54.93	55.28	54.66	56.17	54.02
NETS-ImpGAN (Single)	39.42	38.97	39.08	38.77	39.01	38.32
NETS-ImpGAN (Multiple)	**38.02**	**37.62**	**39.03**	**38.72**	**38.80**	**37.17**
NETS-ImpGAN*			**33.23**			

(b) MV

	DCRNN	Graph WaveNet	IGNNK	GSTAE	GENIE	GTA U-Net
Zero	69.75	67.76	68.13	67.79	67.65	66.45
Mean	69.16	66.67	66.53	66.89	66.77	65.42
NA	66.79	65.23	65.17	64.29	64.88	62.01
TLI	63.35	60.97	61.73	60.98	61.13	60.02
DCMF	48.43	47.03	47.14	47.20	47.29	46.24
TRMF	60.75	59.25	60.69	59.16	59.33	57.20
BRITS	56.67	55.57	56.75	55.23	55.39	54.45
E^2GAN	64.03	64.46	65.23	63.59	63.29	60.88
WDGTC	47.10	44.95	46.11	44.47	44.91	44.26
S-MKKM	56.93	55.38	56.48	55.32	56.25	55.70
GRIN	46.58	44.56	45.73	44.97	44.95	44.36
CSDI	48.35	47.62	49.03	48.01	48.95	46.80
SD-ADMM	59.71	59.24	60.44	59.37	59.88	58.69
mSSA	57.61	57.30	59.42	57.98	58.78	56.25
NETS-ImpGAN (Single)	45.26	44.27	45.61	44.19	44.89	43.95
NETS-ImpGAN (Multiple)	**43.90**	**43.72**	**44.50**	**43.27**	**44.10**	**42.81**
NETS-ImpGAN*			**38.71**			

Note that the combination NETS-ImpGAN plus GTA U-Net have the same capability of capturing inter-time-series and temporal correlations as NETS-ImpGAN*, but the latter allows training with incomplete data in an end-to-end manner and hence avoids error accumulation in two-phase methods.

Table 7. NETS-ImpGAN vs. Ablated Variants on Metro under 25% Missing Rate

	-w/o-T-Conv	-w/o-GAT	-w/o-SA	-w/o-Atten	full NETS-ImpGAN
Random	140.97	43.37	38.86	41.44	**33.23**
MV	84.12	48.59	40.98	43.22	**38.71**

Results are without refinement.

Table 8. ImpGAN vs. Alternative Frameworks on Metro under 25% Missing Rate

	NETS-GAIN	NETS-MisGAN	NETS-Partial VAE	NETS-MIWAE	NETS-P-BiGAN	NETS-ImpGAN
Random	44.56	38.95	40.60	42.71	38.85	**33.23**
MV	52.18	46.77	49.24	45.98	43.07	**38.71**

Results are without refinement.

5.4 Efficacy Study

5.4.1 Ablation Study. We consider the following ablated variants.

— NETS-ImpGAN-w/o-GAT. We remove all the GAT modules in GTANs.
— NETS-ImpGAN-w/o-SA. We remove all the Multi-Head Self-Attention modules in GTANs.
— NETS-ImpGAN-w/o-T-Conv. We remove all the T-Conv modules in GTANs.
— NETS-ImpGAN-w/o-Atten. We remove all the attention mechanisms. We replace GAT by GraphConv [43], which is designed for graphs with static weights. We set the edge weights for GraphConv [43] as follows. We retain the transition probability and processed distance as edge weight for Metro and Air, respectively, and set all edge weights to be 1 for Electricity and Speed, which is similar to the graph construction in previous works [12, 33, 44, 64]. We also remove all the Multi-Head Self-Attention modules.

Table 7 shows the MAE for prediction on Metro under 25% missing rate. NETS-ImpGAN achieves the best performance. Compared to the variants without GAT, Multi-Head Self-Attention, T-Conv, and attention mechanism, the full model NETS-ImpGAN on average reaches 23%, 8%, 65%, and 15% improvement respectively, which affirms the efficacy of these components. Note the particular importance of the T-Conv, without which the performance deteriorates significantly.

5.4.2 Comparison with Alternative Frameworks. We study the efficacy the ImpGAN framework by comparing ImpGAN with alternative frameworks, including GAIN [61], MisGAN [31], Partial VAE [38], MIWAE [40], and P-BiGAN [32]. For a fair comparison, we use the same GTANs inside these frameworks, so that they have the same capability of capturing inter-time-series and temporal correlations as NETS-ImpGAN. The resulted models are denoted by prefix "NETS-".

Table 8 shows the MAE for prediction on Metro under 25% missing rate. The results show that ImpGAN outperforms the other frameworks, which demonstrates the efficacy of ImpGAN.

We further show that ImpGAN avoids the issue of error accumulation in MisGAN and is better suited for imputation/prediction. We compare NETS-ImpGAN with NETS-MisGAN and two additional variants as follows.

— NETS-MisGAN-RealData. It trains the imputer of MisGAN using real complete samples instead of generated ones, which is also equivalent to training NETS-ImpGAN with zero missing rate.
— NETS-ImpGAN-RealMask. It uses real mask samples in place of those generated by G_m.

Table 9 shows the MAE for prediction on Metro under 25% missing rate. Note that NETS-ImpGAN outperforms NETS-MisGAN by a large margin. The large performance gap between

Table 9. NETS-ImpGAN vs. NETS-MisGAN on Metro under 25% Missing Rate

	NETS-MisGAN	NETS-MisGAN-RealData	NETS-ImpGAN	NETS-ImpGAN-RealMask
Random	38.95	29.44	33.23	33.32
MV	46.77	31.31	38.71	38.52

Results are without refinement.

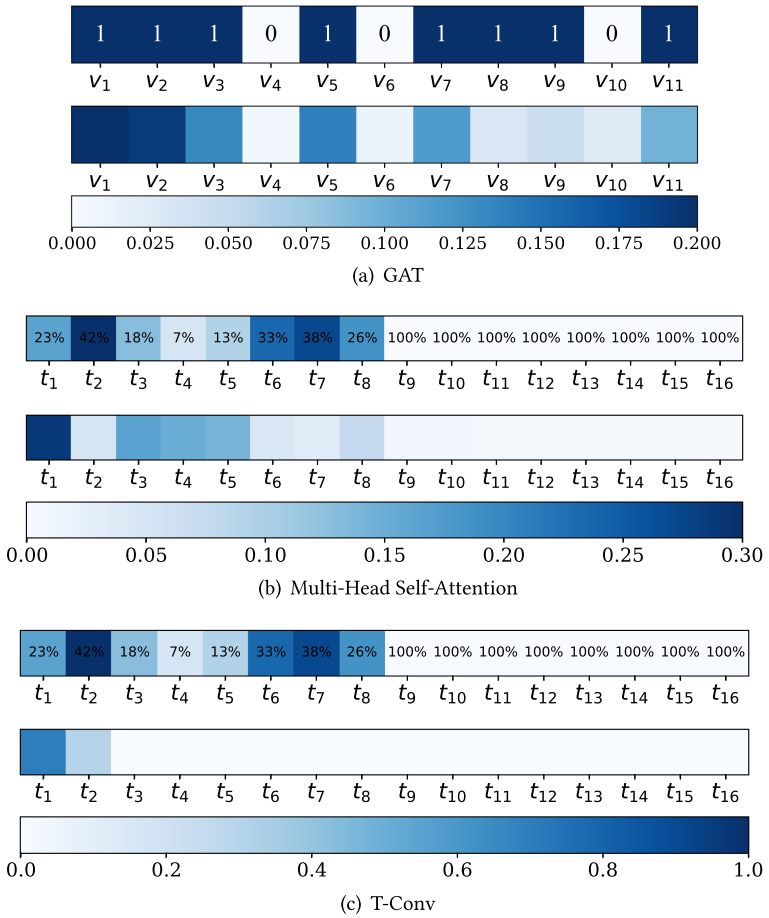
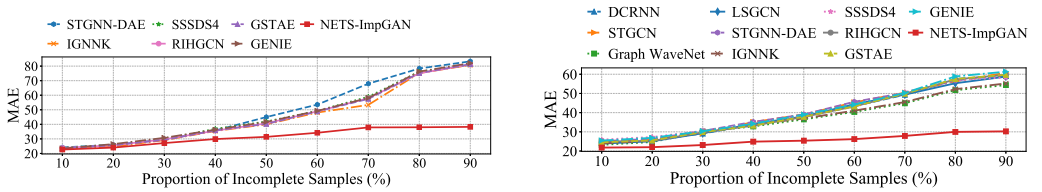


Fig. 8. Visualization of GTAN components.

NETS-MisGAN-RealData and NETS-MisGAN indicates significant error accumulation in MisGAN due to the imperfectly learned joint data distribution. This issue is avoided by NETS-ImpGAN. The imperfectly learned mask distribution can also cause error accumulation, but a comparison between NETS-ImpGAN and NETS-ImpGAN-RealMask shows this effect is negligible. This is because it is easy to learn the binary mask distribution, whose samples are fully observed.

5.4.3 Visualization of GTAN Components. In this section, we give illustrative examples of GAT, Multi-Head Self-Attention, and T-Conv, respectively.

Figure 8(a) visualizes the learned attention coefficients of GAT for one node in a sample. The node has 11 neighboring nodes, denoted by $v_1, \ldots, v_{11}$, the mask of which is indicated by the first line. The corresponding attention coefficients of these nodes are shown in the second line, with



(a) Removing incomplete samples (25% missing rate). (b) Zero imputation (2% missing rate).

Fig. 9. Performance on partially complete Metro dataset under MV.

deeper color representing larger coefficient. We can observe that more attention are assigned to nodes with observed values, which suggests that the attention mechanism helps differentiating the observed values and noise.

Figure 8(b) visualizes an example of learned attention coefficients of self attention. Note that a sample consist of 16 timestamps; for each of the first 8 historical timestamps, some nodes are observed and the others are missing, and the last 8 future timestamps are all treated as missing. In this example, we show the attention paid by the first timestamp in the output to the 16 timestamps in the input. The missing rate of each timestamp is indicated by the first line, and the corresponding attention coefficients are shown in the second line. We can observe that more attention are assigned to timestamps with lower missing rate, except that the highest attention is assigned to the first timestamp in the input, since a timestamp usually has very strong correlation with itself. The visualization also demonstrates that the attention mechanism can help pay more attention to observed values.

Figure 8(c) visualizes the normalized learned weights of the kernel of size 3 in the first T-Conv layer. In this example, we show the kernel corresponding to the first timestamp in the output. Since T-Conv captures temporal correlations from a local view, this first output timestamp only depends on the first and second input timestamp. The weights at t_1 and t_2 are greater than zero, while the others are zero. In contrast, all the attention coefficients in Figure 8(b) are greater than zero.

5.5 Efficiency Study

5.5.1 Data Efficiency. Given a dataset consisting of both complete and incomplete samples, one way to apply methods that require complete data for training is to simply remove all the incomplete samples, which could lead to low data efficiency as mentioned in Section 1. In this section, we take a closer look at this issue. We will refer to methods that require complete data for training simply as complete-data methods. We mainly focus on the following complete-data methods that do not require complete input: STGNN-DAE, IGNNK, SSSDS4, RIHGCN, GSTAE, and GENIE. We do not consider those requiring complete input to exclude the influence of zero-imputed history on the performance when they are fed with incomplete history. For the experiment, we manually introduce missing values in a training set consisting of only complete samples initially, and vary the proportion of incomplete samples. We train NETS-ImpGAN with the entire training set, consisting of both complete and incomplete samples, and train the complete-data methods with only the complete samples. Note that for complete-data methods, the complete samples are used as ground truth, and the actual training inputs are generated by randomly masking the ground truth using the target missing pattern and missing rate, both assumed to be known. All the methods are tested under the same missing pattern and missing rate as the incomplete samples.

Figure 9(a) shows the results on Metro, where the incomplete part follows MV and 25% missing rate and the proportion of incomplete samples varies from 10% and 90% at a step of 10%. When the proportion of incomplete samples is small, the complete-data methods and NETS-ImpGAN

Table 10. Time Cost on Metro under MV and 25% Missing Rate

	DCMF	TRMF	BRITS	E^{2}GAN	GRIN	CSDI	mSSA	NETS-ImpGAN
Training Time (hours)	0	0	1.12	1.25	8.16	2.33	0	9.85
Testing Time (seconds/sample)	0.49	0.37	0.32	1.08	0.26	0.24	0.52	0.19

have similar performance, as there are enough complete samples. As the proportion of incomplete samples increases, the performance gap between the complete-data methods and NETS-ImpGAN becomes larger, as only NETS-ImpGAN can exploit the incomplete samples. This shows that removing incomplete samples can lead to low data efficiency and poor performance.

Another common practice to exploit incomplete samples in complete-data methods is to first impute the missing entries with zeros, and then use both imputed and originally complete samples for training [66]. We also explore the performance of such adaptation on partially complete dataset. Here we additionally consider the following methods that require complete input: DCRNN, STGCN, Graph WaveNet, and LSGCN. Figure 9(b) shows the results under the same setting as in Figure 9(a), except that we set the missing rate of the incomplete part to the very low level of 2% in favor of zero imputation. The training procedure is also the same, except that the ground truth set is now obtained by zero imputing the original incomplete samples. All the methods are tested under MV and 2% missing rate, and we also use zero imputation for those requiring complete input. While zero imputation improves on removing incomplete samples, the gap between the complete-data methods and NETS-ImpGAN still becomes large with the increase of the incomplete proportion.

5.5.2 Model Efficiency. In this section, we compare the efficiency of NETS-ImpGAN with that of the baselines in terms of time cost and model complexity. Table 10 shows the training time and testing time per sample on Metro under MV and 25% missing rate. Mean, TLE, and LO have very short testing time but in general poor performance, and thus we omit them in the table.

Among these methods, DCMF, TRMF, and mSSA do not need training but require more time than most of the others for each testing sample, respectively. In contrast, deep learning methods, including BRITS, E^{2}GAN, GRIN, CSDI, and NETS-ImpGAN, require a long time for training but typically less time for testing, except for E^{2}GAN, which takes even more time than DCMF, TRMF, and mSSA for testing. Compared to BRITS and GRIN that involve recurrent computation, our NETS-ImpGAN uses convolution modules that enjoy concurrent computation, leading to even faster testing process. From the time cost perspective, NETS-ImpGAN is better suited for scenarios where the model does not need to be updated frequently and the prediction accuracy is more important, while DCMF, TRMF, and mSSA are better suited for scenarios where the model needs to be updated frequently but is used to predict only a small number of times after each update.

In terms of model complexity, DCMF, TRMF, and mSSA have a small number of parameters, as the former assumes a somewhat restrictive linear system model, and the latter is based on matrix factorization. Among the deep learning methods, BRITS, E^{2}GAN, and CSDI have fewer parameters than NETS-ImpGAN, as the former three only have temporal modules. GRIN and NETS-ImpGAN have a similar complexity. Note that none of the smaller models can increase their number of parameters in a straightforward way to improve their performance.

6 CONCLUSION

In this article, we study the *prediction of NETS with incomplete data*. We propose *NETS-ImpGAN*, a novel deep learning framework for both prediction and imputation that can be trained on incomplete data with missing values in both history and future. We design *Graph Temporal Attention Networks* that can adapt to different samples when capturing the inter-time-series

and temporal correlations. We conduct extensive experiments on four real-world datasets under different missing patterns and missing rates. The results show that NETS-ImpGAN outperforms existing methods, reducing the MAE of the second best up to 25%.

REFERENCES

[1] Anish Agarwal, Abdullah Alomar, and Devavrat Shah. 2022. On multivariate singular spectrum analysis and its variants. In *Abstract Proceedings of ACM SIGMETRICS PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems*. 79–80.

[2] Juan Miguel Lopez Alcaraz and Nils Strodthoff. 2023. Diffusion-based time series imputation and forecasting with structured state space models. *Trans. Mach. Learn. Res.* (2023), 1–36.

[3] Paul D. Allison. 2001. *Missing Data*. Sage Publications, USA.

[4] Irish Social Science Data Archive. 2016. ISSDA | Commission for Energy Regulation (CER). Retrieved from https://www.ucd.ie/issda/data/commissionforenergyregulationcer/

[5] Martin Arjovsky, Soumith Chintala, and Leon Bottou. 2017. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*. 214–223.

[6] Lei Bai, Lina Yao, Salil S. Kanhere, Xianzhi Wang, and Quan Z. Sheng. 2019. STG2Seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 1981–1987.

[7] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. 2016. Importance weighted autoencoders. In *Proceedings of the 4th International Conference on Learning Representations*.

[8] Yongjie Cai, Hanghang Tong, Wei Fan, and Ping Ji. 2015. Fast mining of a network of coevolving time series. In *Proceedings of the SIAM International Conference on Data Mining*. 298–306.

[9] Yongjie Cai, Hanghang Tong, Wei Fan, Ping Ji, and Qing He. 2015. Facets: Fast comprehensive mining of coevolving high-order time series. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 79–88.

[10] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. 2018. BRITS: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems*. 6775–6785.

[11] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1724–1734.

[12] Andrea Cini, Ivan Marisca, and Cesare Alippi. 2022. Filling the gaps: Multivariate time series imputation by graph neural networks. In *International Conference on Learning Representations*.

[13] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*. 933–941.

[14] DiDiChuxing. 2018. City Traffic Index. Retrieved from https://gaia.didichuxing.com/

[15] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. 2022. GENIE: Higher-order denoising diffusion solvers. In *Advances in Neural Information Processing Systems*. New Orleans, USA. 30150–30166.

[16] Jeff Donahue, Philipp Krahenbuhl, and Trevor Darrell. 2017. Adversarial feature learning. In *International Conference on Learning Representations*.

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

[18] Yongshun Gong, Zhibin Li, Jian Zhang, Wei Liu, Bei Chen, and Xiangjun Dong. 2020. A spatial missing value imputation method for multi-view urban statistical data. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. 1310–1316.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2672–2680.

[20] Albert Gu, Karan Goel, and Christopher Re. 2022. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

[21] Hairi, Hanghang Tong, and Lei Ying. 2019. NetDyna: Mining networked coevolving time series with missing values. In *Proceedings of the IEEE International Conference on Big Data*. 503–512.

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

[23] Rongzhou Huang, Chuyin Huang, Yubao Liu, Genan Dai, and Weiyang Kong. 2020. LSGCN: Long short-term traffic prediction with graph convolutional networks. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. 2355–2361.

[24] Baoyu Jing, Hanghang Tong, and Yada Zhu. 2021. Network of tensor time series. In *Proceedings of the Web Conference*. 2425–2437.
[25] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*.
[26] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.
[27] Roger Koenker and Kevin F. Hallock. 2001. Quantile regression. *J. Econ. Perspect.* 15, 4 (2001), 143–156.
[28] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2021. DiffWave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*.
[29] Sanmukh R. Kuppannagari, Yao Fu, Chung Ming Chueng, and Viktor K. Prasanna. 2021. Spatio-temporal missing data imputation for smart power grids. In *Proceedings of the 12th ACM International Conference on Future Energy Systems*. 458–465.
[30] Trent Kyono, Yao Zhang, Alexis Bellot, and Mihaela van der Schaar. 2021. MIRACLE: Causally-aware imputation via learning missing data mechanisms. In *Advances in Neural Information Processing Systems*. 23806–23817.
[31] Steven Cheng-Xian Li, Bo Jiang, and Benjamin Marlin. 2019. Learning from incomplete data with generative adversarial networks. In *Proceedings of the 7th International Conference on Learning Representations*.
[32] Steven Cheng-Xian Li and Benjamin Marlin. 2020. Learning from irregularly-sampled time series: A missing data perspective. In *Proceedings of the 37th International Conference on Machine Learning*. 5937–5946.
[33] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *Proceedings of the 6th International Conference on Learning Representations*.
[34] Ziyue Li, Nurettin Dorukhan Sergin, Hao Yan, Chen Zhang, and Fugee Tsung. 2020. Tensor completion for weakly-dependent data on graph for metro passenger flow prediction. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. 4804–4810.
[35] Roderick J. A. Little and Donald B. Rubin. 1986. *Statistical Analysis with Missing Data*. John Wiley & Sons, New York, NY.
[36] Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, and Xiaojie Yuan. 2018. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*. 1596–1607.
[37] Yonghong Luo, Ying Zhang, Xiangrui Cai, and Xiaojie Yuan. 2019. E^2GAN: End-to-end generative adversarial network for multivariate time series imputation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 3094–3100.
[38] Chao Ma, Sebastian Tschiatschek, Konstantina Palla, Jose Miguel Hernandez-Lobato, Sebastian Nowozin, and Cheng Zhang. 2019. EDDI: Efficient dynamic discovery of high-value information with partial VAE. In *Proceedings of the 36th International Conference on Machine Learning*. 4234–4243.
[39] Ivan Marisca, Andrea Cini, and Cesare Alippi. 2022. Learning to reconstruct missing data from spatiotemporal graphs with sparse observations. In *Advances in Neural Information Processing Systems*. 32069–32082.
[40] Pierre-Alexandre Mattei and Jes Frellsen. 2019. MIWAE: Deep generative modelling and imputation of incomplete data sets. In *Proceedings of the 36th International Conference on Machine Learning*. 4413–4423.
[41] Bennet E. Meyers and Stephen P. Boyd. 2023. Signal decomposition using masked proximal operators. *Found. Trends Sign. Process.* 17, 1 (2023), 1–78.
[42] Microsoft. 2012. Urban Air—Microsoft Research. Retrieved from https://www.microsoft.com/en-us/research/project/urban-air/
[43] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. 4602–4609.
[44] Junjie Ou, Jiahui Sun, Yichen Zhu, Haiming Jin, Yijuan Liu, Fan Zhang, Jianqiang Huang, and Xinbing Wang. 2020. STP-TrellisNets: Spatial-temporal parallel trellisnets for metro station passenger flow prediction. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. 1185–1194.
[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*. 234–241.
[46] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323 (1986), 1476–4687. Issue 6088.
[47] Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Sign. Process.* 45, 11 (1997), 2673–2681.
[48] Feiyang Sun, Pinghui Wang, Junzhou Zhao, Nuo Xu, Juxiang Zeng, Jing Tao, Kaikai Song, Chao Deng, John C. S. Lui, and Xiaohong Guan. 2022. Mobile data traffic prediction by exploiting time-evolving user mobility patterns. *IEEE Trans. Mob. Comput.* 21, 12 (2022), 4456–4470.

[49] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*. 24804–24816.
[50] TianChi. 2019. TianChi Global Urban Computing AI Competition: Metro Passenger Flow Prediction. Retrieved from https://tianchi.aliyun.com/competition/entrance/231708/introduction/
[51] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A generative model for raw audio. In *Proceedings of the 9th ISCA Speech Synthesis Workshop*. 125–125.
[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.
[53] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.
[54] Alexander H. Waibel, Toshiyuki Hanazawa, Geoffrey E. Hinton, Kiyohiro Shikano, and Kevin J. Lang. 1989. Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoust. Speech Sign. Process.* 37, 3 (1989), 328–339.
[55] Ao Wang, Yongchao Ye, Xiaozhuang Song, Shiyao Zhang, and James J. Q. Yu. 2023. Traffic prediction with missing data: A multi-task learning approach. *IEEE Trans. Intell. Transport. Syst.* 24, 4 (2023), 4189–4202.
[56] Yuankai Wu, Dingyi Zhuang, Aurelie Labbe, and Lijun Sun. 2021. Inductive graph neural networks for spatiotemporal kriging. *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. 4478–4485.
[57] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 1907–1913.
[58] Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Q. Weinberger. 2018. An empirical study on evaluation metrics of generative adversarial networks. arXiv1806.07755. Retrieved from https://arxiv.org/abs/1806.07755
[59] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. 2019. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. 5668–5675.
[60] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 2588–2595.
[61] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GAIN: Missing data imputation using generative adversarial nets. In *Proceedings of the 35th International Conference on Machine Learning*. 5675–5684.
[62] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3634–3640.
[63] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S. Dhillon. 2016. Temporal regularized matrix factorization for high-dimensional time series prediction. In *Advances in Neural Information Processing Systems*. 847–855.
[64] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. 1655–1661.
[65] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, and Xiuwen Yi. 2016. DNN-based prediction model for spatio-temporal data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*.
[66] Weida Zhong, Qiuling Suo, Xiaowei Jia, Aidong Zhang, and Lu Su. 2021. Heterogeneous spatio-temporal graph convolution network for traffic forecasting with missing values. In *Proceedings of the 41st IEEE International Conference on Distributed Computing Systems*. 707–717.

Received 12 May 2023; revised 28 October 2023; accepted 16 January 2024