Information Sciences 643 (2023) 119234

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Time series imputation with GAN inversion and decay connection

Longfei Xu *, Lingyu Xu, Jie Yu

School of Computer Engineering and Science, Shanghai University, Shanghai, China

ARTICLE INFO

Keywords:
Time series imputation
Generative adversarial networks
Graph convolutional networks
GAN inversion
Time decay

ABSTRACT

Time series imputation is essential for real-world applications. Though the emergence of Generative Adversarial Networks (GANs) and Graph Convolution Networks (GCNs) provides more possibilities to improve imputation performance, how to achieve the optimal latent code and precisely model the properties of incomplete time series remain a challenge. In GAN-based methods, an effective latent code of incomplete time series is necessary for precise reconstruction. To acquire the optimal latent code, we introduce GAN inversion to invert the input to the latent space of a pretrained GAN. The inverted latent code contains rich properties of original observations and thus can better reconstruct the target sample. To model the temporal irregularity due to the presence of missing values, the decay connection is exploited to quantify the influence that dependencies between adjacent observations should decrease as the time lags between them increase. We incorporate the quantification into the adjacent matrix of the GCN to better aggregate adjacent information of incomplete time series. With the adoption of decay connection, the resulting latent code through GAN inversion can further produce faithful reconstruction. Quantitative and qualitative experiments conducted on several time series datasets show that our proposal achieves state-of-the-art or competitive imputation performance.

1. Introduction

Time series are important data resources in diverse fields. Complete and detailed datasets benefit the comprehensive exploration of related tasks. However, missing values in time series are ubiquitous due to sensor failures, software malfunctions and human errors, etc [1,2]. The corrupted data will usually cause ambiguity in pattern analysis and thus deteriorate the performance of downstream applications.

To address the problem, extensive methods have been proposed to handle incomplete time series. Deletion [3] is a traditional method that directly discards samples with missing values while ignores the correlation between samples. This method will squander valuable observed values and cause significant information loss. In contrast, imputation is a more appropriate scheme which estimates missing values through existing observations. Mean and median are commonly used imputation methods which take advantage of statistical characteristics of time series [4]. They replace missing values with mean or median values of observed data. In spite of simplicity, they tend to neglect the variation of data and fail to utilize the inherent temporal dependencies of time series, which will result in poor imputation performance.

The machine learning is another available option for the problem. KNN [5] looks for samples similar to the given samples by distance measurements, which inevitably goes through the whole dataset to find target samples. The expensive computation limits its

* Corresponding author at: School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China.
E-mail address: xulongfei@shu.edu.cn (L. Xu).

https://doi.org/10.1016/j.ins.2023.119234
Received 15 January 2023; Received in revised form 2 May 2023; Accepted 23 May 2023
Available online 29 May 2023

application given large datasets. EM [6] algorithm iteratively updates parameters and imputes missing values through calculating maximum likelihood estimation while suffers from slow convergence. MICE [7] is a multiple imputation method and takes into account the statistical uncertainty of missing values. MICE trains a series of prediction models for different features and produces multiple alternative imputation values for each missing value conditioned on the observed parts. Its extension [8] further account for and quantify the imputation uncertainty with active learning. These methods usually impose too many assumptions on the properties of incomplete time series or require large computation resources. The defects make them less practical in real imputation scenarios.

The rapid emergency of deep learning brings more possibilities to effectively impute missing values in time series. The general trend for time series imputation by deep learning is to fully excavate the temporal dependencies and feature correlations of data. RNN [9] aims to memorize temporal dependencies with special recursive units, the variants of which have been proposed to handle missing value imputation. Relevant works usually modify the recursive units for adaption to missing patterns or add bidirectional structures to find multiple temporal dependencies [10]. As a typical variant of RNN, GRU-D [11] uses a mask to indicate whether data are missing as a part of the input and then trains a model with the masked data. It finds that the missing pattern is informative and utilizes a decay mechanism to capture correlations between missing patterns and data labels. BRITS [12] extends the mechanism to the recurrent dynamical system and estimates missing values with downstream tasks. To unearth extra information from different perspectives, [13] constructs the convolutional kernel to employ the feature correlations by learning the non-linearity and spatial arrangement of given data. The data is grouped into different clusters to further capture correlations between features.

Recently, the widespread use of GANs [14] further promotes the improvement of imputation performance. GAN is proposed to deal with image synthesis tasks by mapping the distribution of random variables to that of real images via adversarial training. Its strong ability of generating realistic is migrated to tabular data and time series imputation [15-17] and shows extraordinary application prospects. More recently, GCN [18] receives considerable attention because of the aggregation of adjacent information. It is proposed to model complicated relationships among graph-structured data. Its representation learning ability on graphs is applied to missing values imputation [19-21].

Though previously mentioned approaches effectively impute missing values, few of them try to make specific optimizations according to the characteristics of incomplete time series. In the paper, we design the decay connection to quantify irregular temporal dependencies of incomplete time series and incorporate the quantization into the GCN. To recover original data, we choose GAN inversion [22,23] to map incomplete data into a low-dimensional latent code. GAN inversion aims to find a optimal latent code in the latent space of a pretrained GAN to reconstruct input. To complete the mapping, the model is supposed to characterize the properties of input objects. To this end, we propose to combine GAN inversion with decay connection in the GCN to recover original time series. GAN inversion currently is applied to tasks such as point cloud completion [24,25] and image inpainting [26] and shows state-of-the-art performance. To the best of our knowledge, it is the first time that GAN inversion is extended to time series imputation with GCN. The main contributions of this paper can be summarized as follows:

- We introduce GAN inversion into GCN to encode the input time series to the learned latent space of a pretrained GAN. The inferred latent code contains rich properties and provides constraints of original data space, which will exert multiple reasonable imputation results.
- To model the temporal irregularity of incomplete time series, we exploit the decay connection to quantify adjacent properties between samples. The quantization is incorporated into the adjacent matrix of the GCN to better aggregate adjacent information and generate data more effectively conditioned on partial supervision.
- Qualitative and quantitative experiments on several time series datasets suggest that our proposal achieves state-of-the-art or competitive imputation performance.

The remains of this paper are organized as follows. We describe relevant background technologies in Section 2 and introduce preliminaries in Section 3. Section 4 elaborates the structure of our proposed model in detail. Section 5 presents the experimental results and analysis and we conclude our work in Section 6.

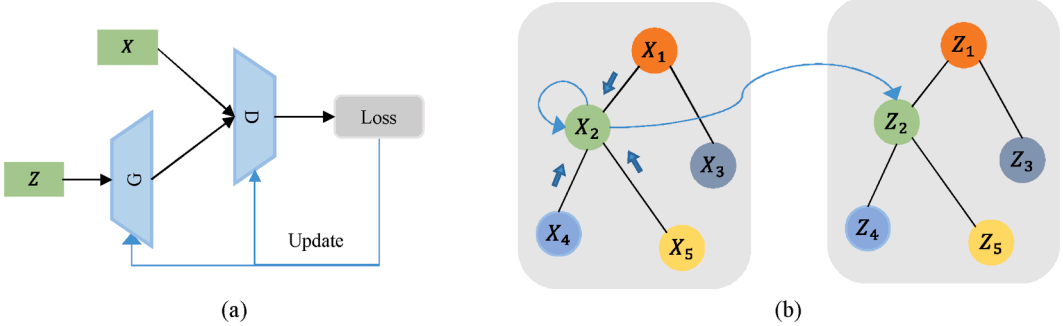

Fig. 1. Schematic Diagram of GAN(a) and GCN(b).

2. Related work

2.1. Generative imputation

GANs are proposed to continuously improves the generation performance of generative methods via adversarial training. The vanilla GAN is composed of a generator G and a discriminator D to jointly play a min–max game. The structure is shown in Fig. 1(a). Due to its strong ability to generate realistic samples, GAN is widely used for image process such as generation [14] and inpainting [26]. Therefore, some representative works including GAN-2-Stage [15], E^2GAN [16] and GAIN [17] investigate its extension to tabular data and time series imputation. GAN-2-Stage employs the modified GRUI as the adversarial component. This composition benefits the extraction of temporal dependencies of time series. GAN-2-Stage first samples a noise vector from a random distribution as input to train the model. Then an extra stage is needed to optimize the noise vector through the trained model to infer reasonable values. Though the two-stage training scheme generates effective target samples, the time-consuming problem poses a challenge. To avoid the optimization stage and promote time efficiency, E^2GAN designs an end-to-end framework where an encoder is introduced to map the incomplete input to the deserved optimized noise vector with little deterioration of imputation performance. However, the inherent defect of unstable training of GANs still exists in these imputation methods. The subsequent work [27] proposes a real-data forcing method which combines real and generated data in the generator to stabilize the training process. This modification also prevents the generation error at the current time from worsening the later imputed values. Different from the aforementioned methods which explore the noise vector to recover target samples, GAIN focus on the establishment of a hint mechanism to generate the real data distribution. This mechanism provides extra information of original samples to help the discriminator better distinguish the observed components from the imputed ones. PC-GAIN [28] and SSGAN [29] are two variants of GAIN proposed to address unsupervised and semi-supervision scenarios. PC-GAIN uses clustering algorithm to get pseudo labels for all the unlabeled samples. SSGAN trains a classifier with partial labeled samples and predicts labels for other unlabeled samples. What they have in common is that they all exploit the auxiliary classifier as a constraint for the generator to generate samples that meet the conditions. [30] also employs the hint mechanism to learn the missing pattern and extends to the conditional GAN framework. Its main contribution lies in that the temporal dependencies and feature correlations of given data are incorporated into the modified RNN to estimate missing values. The denoising diffusion probabilistic model [31] is a class of generative model which gradually adds noise to input in the forward process defined by a Markov chain and converts noisy data into plausible input in the backward process. This model has made great achievements in image synthesis, restoration and other fields. CSDI [32] extends the diffusion model to time series imputation and learns conditional distribution by taking observed values as a conditional input. SSSD [33] exploits the state-space models as building blocks of the modified diffusion model architecture to capture temporal dependencies for time series imputation and forecasting. The discussed methods can not always get the optimal latent code or suffer from expensive computing, which may not perform well in real imputation scenarios.

2.2. Graph imputation

Non-Euclidean data including graph-structured data, ranging from social networks to chemical compounds, are ubiquitous and contains rich relational and structural properties. Convolutional neural networks specialize in Euclidean data while show poor performance faced with non-Euclidean data. To this end, GCN is proposed to effectively model complicated relationships among graph-structured data. GCN can handle non-Euclidean data with complex topological structure and extends the convolution operation into irregular regions. It combines spectral and spatial operations and employs the linear approximation of convolution kernels via Chebyshev polynomials. The structure is shown in Fig. 1(b). For data imputation, GCN is also an appropriate option. To address graphs with missing values, GINN [19] builds a graph autoencoder in an adversarial framework and takes into account the global and local properties to infer missing values. GCNMF [20] incorporates the imputation process and graph learning within the same framework. The incomplete features are transformed to follow the Gaussian mixture distribution and derives the expected activation of neurons in the first layer of GCN. GRAPE [34] conducts feature imputation and label prediction using a graph representation method. By creating a bipartite graph structure, it introduces the edge embeddings and augmented node features for data imputation. These methods focus on tabular data and are not skilled in time series. Lately, to take advantage of the strong relational information existing within interconnected sensors, GRIN [21] reconstructs incomplete time series by learning spatio-temporal representations coupled with RNN and GCN. However, none of them emphasize the temporal irregularity or try to find the corresponding latent code of incomplete time series. Hence, we propose an imputation method to infer the optimal latent code and characterize the irregular property in GCN.

3. Preliminaries

3.1. Problem formulation

We denote an actually observed data as $X = (x_0, \cdots, x_{n-1}) \in \mathbb{R}^{n\times d}$ with d different time series, which are observed at timestamps $T = (t_0, \cdots, t_{n-1})$. X is generally a matrix where each column $X^d = (x_0^d, \cdots, x_{n-1}^d)$ denotes the d-th time series and each row $X_t = (x_t^0, \cdots, x_t^{d-1})$ is the observed sample at timestamps t. For simplicity, we denote X_t^j as the j-th feature value of X at timestamp t_i. We randomly drop some elements on a complete observation as the input. The mask matrix $M \in \mathbb{R}^{n\times d}$ that records the missing positions and takes values in $\{0,1\}^d$ is defined as

$$M_i^j = \begin{cases} 1, & \text{if } X_i^j \text{ is observed} \\ 0, & \text{if } X_i^j \text{ is missing} \end{cases}. \tag{1}$$

Then the corrupted input data X^{inc} can be represented by

$$X^{inc} = X \odot M, \tag{2}$$

where $\odot$ is the element-level multiplication. The goal of the time series imputation is to find an appropriate value for each missing component in X^{inc} to minimize the error between the reconstructions and actual observations.

3.2. Adversarial training

In the vanilla GAN, G receives a random vector Z sampled from a latent space that conform a specific distribution to produce highly realistic samples. D takes as input the real or fake samples to distinguish between them. G and D compete with each other to adjust the model performance. The main purpose of the vanilla GAN is to learn the mapping from the input distribution to the target distribution and produce indistinguishable realistic samples. The object function consists of

$$L_G = \mathbb{E}_{Z\sim P_Z}[-\mathrm{D}(\mathrm{G}(Z))], \tag{3}$$

$$L_D = \mathbb{E}_{Z\sim P_Z}[\mathrm{D}(\mathrm{G}(Z))] - \mathbb{E}_{X\sim P_X}[\mathrm{D}(X)], \tag{4}$$

where P_z and P_x respectively represent the distributions of the random vector and real samples. $\mathrm{D}(X)$ is the probability that the real sample X is real and $\mathrm{D}(\mathrm{G}(Z))$ represents the probability that the generated sample $\mathrm{G}(Z)$ is real. $\mathrm{D}(X)$ takes values in $[0,1]$, where 1 and 0 mean that the sample is total real or fake. The ideal state comes with $\mathrm{D}(\mathrm{G}(Z)) = 0.5$, which means G generates highly authentic samples and D can not distinguish them from true samples.

3.3. Graph convolution

An undirected graph $G = (V, E)$ consists of nodes $V = \{\nu_i | i = 0, 1, 2, \ldots N-1\}$ and edges $E = \{e_{ij} | i, j = 0, 1, 2, \ldots N-1\}$, where N denotes the amount of nodes in the graph. e_{ij} represents the edge formed by ν_i and ν_j, which can be regarded as the spatial connection between the corresponding nodes. The feature matrix of nodes in a graph is denoted by $X \in \mathbb{R}^{N\times D}$ where D is the dimension of each node. X_{ij} is the j-th features of node ν_i. The graph is utilized to integrate all the nodes into a graph structure with the adjacency matrix $\mathrm{A} \in \mathbb{R}^{N\times N}$. A_{ij} refers to the effect that ν_i has on ν_j and $A_{ij} = A_{ji}$. For the model with GCN of l layers, we have

$$H^{l+1} = \sigma\left(LH^l W^l\right), \tag{5}$$

where $H^l \in \mathbb{R}^{\mathrm{N}\times D^l}$ and $W^l \in \mathbb{R}^{D^l \times D^{l+1}}$ respectively denote the aggregation of l-th layer and learnable parameters. For the first layer, we set $H^0 = X$. $L = \widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2} \in \mathbb{R}^{N\times N}$ is the renormalized graph Laplacian matrix of $\widetilde{A} = A + I$ with self-loops and I is the identity matrix. $\widetilde{D}$ is the degree matrix of $\widetilde{A}$ and $\widetilde{D}_{ii} = \sum_j^N \widetilde{A}_{ij}$. $\sigma(\cdot)$ refers to the activation function. We use multiple one-layer GCNs to construct

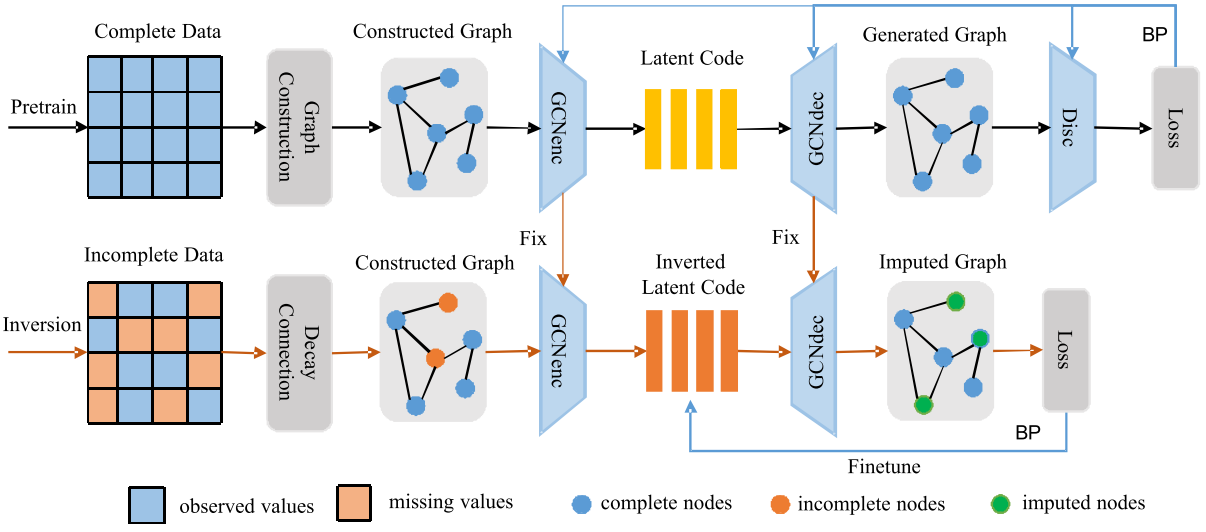

Fig. 2. Framework of the proposed model.

the building block of our proposed model.

4. Methodology

4.1. Overall framework

In this part, we introduce the general framework of the proposed model for time series imputation. The model consists of two main components including the decay connection and GAN inversion. We will respectively show details in the following sections. The decay connection is exploited to construct the graph of incomplete time series to model the properties of temporal irregularity. The GAN inversion tries to encode the incomplete observation into a low-dimensional latent code Z with an optimization process. The optimized latent code is fed into a pretrained GCN decoder to impute the incomplete observation. With the help of decay connection, the GAN inversion will derive a more effective embedding to faithfully reconstruct original time series.

The overall framework is briefly depicted in Fig. 2. In the pretraining stage, the complete data X is fed sequentially into the encoder E, decoder (generator) G, and discriminator D which are all based on one-layer GCNs. G and D preserve the adversarial architecture as that in the vanilla GAN. E is trained together with G and D to encode X into a low-dimensional latent code Z. In the contrary, G takes as input the latent code to generate time series $\widehat{X}$, which is then fed into D with X to distinguish the real data from the generated data. The pretraining procedure can be formulated as follows:

$$Z = \mathrm{E}(X, A) = \sigma\left(LXW^{1}\right), \tag{6}$$

$$\widehat{X} = \mathrm{G}(Z, A) = \sigma\left(LZW^{2}\right). \tag{7}$$

When the aforementioned components converge after some epochs, the GAN inversion will intervene in the process. GAN inversion aims to invert a real sample back into the latent space of a well-trained GAN to precisely reconstruct the original sample. To achieve the qualified GAN, the parameters of E, G and D are fixed after some epochs. The well-trained GAN contains rich properties about X and can be regarded as an effective prior for time series imputation. In the inversion stage, we first randomly drop some values from X to simulate real missing scenarios and get the incomplete time series X^{inc}. Then we adopt the decay connection to model the temporal irregularity of X^{inc} owing to the presence of missing values. Then the constructed graph of X^{inc} is fed into the fixed E to get an optimized latent code Z derived from X^{inc}. To some extent, Z can coarsely reconstruct X when fed into the fixed G. However, Z is usually not the optimal and can not fully exploit the rich information in G, which lead to the resulting reconstruction unsatisfactory. To get the optimal latent code Z^{*}, another optimization process is needed. The reconstruction loss between X^{inc} and $\widehat{X}$ from the fixed G is utilized to directly optimize Z from coarse to fine by back propagation [35] with several epochs. With the adoption of the optimization process and decay connection, Z^{*} is achieved and will lead to a precise reconstruction of X. In the next Section 4.2 and 4.3, we will respectively specify the details of the GAN inversion and decay connection.

4.2. GAN inversion

GAN inversion [36] is recently studied to exploit the semantics it learns spontaneously. For a target sample X, GAN inversion aims to infer the appropriate latent code Z from the learned latent space of a pretrained GAN, which when passed through the trained generator produces a reconstruction similar to X. This emerging inference technology has been widely applied to diverse fields, especially the image inpainting [26], editing [37] and translation [38]. Compared to the counterpart, the available literature that describes GAN inversion with time series is meager. Inspired by its success in these tasks, we introduce GAN inversion in the framework of GCN for time series imputation. To the best of our knowledge, it is the first time to incorporate GAN inversion with GCN for time series imputation.

Existing GAN inversion methods can be divided into the optimization method [22,39], encoder method [40,41] and hybrid method [42]. The optimization method directly optimizes the latent code by gradient descent from a random initial state, which can produce samples with high quality. While this method consumes a lot of computing resources and is easy to fall into local optima. To alleviate the defect, the encoder method trains an encoder to get the latent code and jointly optimize the encoder. It is usually faster but with inferior reconstruction quality. Apart from these two methods, the hybrid method combines the advantages of both. It aims to find an

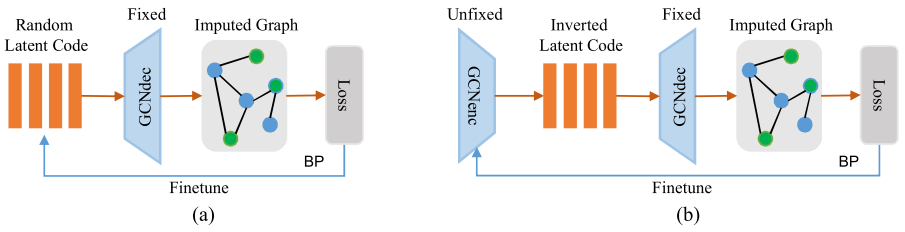

Fig. 3. Optimization (a) and encoder (b) method.

appropriate initial state for the latent code with a trained encoder and then optimize the code from the encoded state.

In some previous study [15,16], some techniques similar to GAN inversion are preliminarily explored for time series imputation. GAN-2-Stage directly optimizes the noise vector sampled from a random distribution to impute missing values. As discussed above, the optimization method takes a lot of time to reconstruct the whole time series. E^2GAN trains an encoder to map the incomplete observation into the low-dimensional embedding, which provides supervision for the generated sample. This method forms an end-to-end framework and saves computing resources. However, the encoded embedding fails to get further optimized, which makes the method struggle to acquire the optimal latent code. What's more, these methods are based on the framework of GRU which can not sufficiently exploit the adjacent properties of time series. In this study, we use the hybrid method to conduct GAN inversion in the framework of GCN.

In the optimization method depicted in Fig. 3(a), the optimal latent code Z^* conforms to the following formula:

$$Z^* = \underset{Z\in\mathbb{R}^d}{\operatorname{argmin}} \ell\left(X^{inc}, \mathrm{G}(Z)\odot M\right), \tag{8}$$

where $\mathbb{R}^d$ denotes the distribution of Z and G is fixed. ℓ is the evaluation criterion. Z^* should ensure the values between X^{inc} and $\mathrm{G}(Z)$ in the observed positions specified by the mask matrix M are as same as possible. This optimization problem is non-convex and heavily depends on the proper initialization of Z, otherwise it is easy to fall into local optima.

The encoder method depicted in Fig. 3(b) converts the problem into optimizing the parameters of an encoder, which indirectly predicts Z^* from X^{inc}. The training objective is defined by:

$$\theta_E^* = \underset{\theta_E}{\operatorname{argmin}}\ \ell\left(X^{inc}, \mathrm{G}\left(\mathrm{E}\left(X^{inc};\theta_E\right)\right)\odot M\right), \tag{9}$$

where the encoder E is parameterized by θ_E. Once the optimal parameters θ_E^* is achieved, $\mathrm{E}(X^{inc};\theta_E^*)$ can be seen as the optimal latent code inferred by the encoder method. This method often performs better and does not fall into local optima. Normally, E and G are pretrained and G is frozen in the training process. Note that the optimal latent code $\mathrm{E}(X^{inc};\theta_E^*)$ by the encoder method usually can not best reconstruct specific single samples. In the hybrid method, the intermediate latent code $\mathrm{E}(X^{inc};\theta_E)$ inferred by the encoder method is regarded as an effective initial state which contains large properties of original samples and further optimized by gradient descent to obtain Z^*. In our experiments, we minimize the L2-norm, i.e., mean squared error (MSE) as ℓ to further optimize $\mathrm{E}(X^{inc};\theta_E)$, yielding the following optimization problem:

$$Z^* = \underset{Z\in\mathbb{R}^d}{\operatorname{argmin}}\left[\mathrm{MSE}\left(X^{inc}, \mathrm{G}(Z)\odot M\right)\right], \tag{10}$$

Where $Z = \mathrm{E}(X^{inc};\theta_E)$. With the pretrained E, Z is landed in the semantic domain of the original samples [23]. What's more, with the further optimization for Z, the derived Z^* can faithfully recover the original input. There are also some methods that simultaneously optimize the intermediate latent code and finetune the generator such as [24,43]. For simplicity, we choose to optimize the intermediate latent code by hybrid method.

4.3. Decay connection

One of the key obstacles in time series imputation is how to effectively model the temporal irregularity of incomplete data [15,16]. GAN-2-stage finds that the time lag is changeable between two existent elements due to the presence of missing values. The larger time lag will yield less influence. It uses the time decay vector to quantify the influence in order to better model the latent dependencies between observations. [30] also puts forward a similar view that there exist strong dependencies between the observed and unobserved parts in incomplete time series and the generated data are ought to be conditioned by the observed parts. Inspired by [15,16,29,44], we propose the time decay adjacent matrix to construct the decay connection between nodes to model the temporal irregularity caused by missing values. The basic idea is that the dependencies between two adjacent observations should decrease as the time lag between them increases. In this way, the influence of missing values can be effectively quantified to better model the temporal irregularity of incomplete time series.

In time series, some elements might be missing during consecutive timestamps. For each time series matrix X, we define a time lag matrix $\delta\in\mathbb{R}^{n\times d}$ to record the time lag between the current and last timestamp with an observed value. The calculation of δ is defined below. For simplicity, we denote δ_i^j as the j-th feature value of δ at t_i:

$$\delta_i^j = \begin{cases} 0, & \text{if } i = 1 \\ t_i - t_{i-1}, & \text{if } M_{i-1}^j = 1 \text{ and } i > 1 \\ \delta_{i-1}^j + t_i - t_{i-1}, & \text{if } M_{i-1}^j = 0 \text{ and } i > 1 \end{cases}. \tag{11}$$

It can be inferred from δ that missing values will exert more influence on the adjacent observed values. Note that δ represents the element-level time lags. While in the GCN, the adjacent matrix $\widetilde{A}$ represents the node-level dependencies. Therefore, it is necessary to transform the element-level representations of time lags into node-level dependencies to model the temporal irregularity of incomplete time series. We introduce the decay connection to construct an undirected graph as follows to reflect the tendency that the node-level

dependencies decrease with the increase of time lags:

$$\widetilde{A_k^h} = \begin{cases} 1, \text{ if } k = h \\ \widetilde{A_h^k}, \text{ if } k > h \\ \dfrac{1}{\sum_{i=k}^{h}\sum_{j=1}^{n}\frac{1}{n}\delta_i^j}, \text{ if } k < h \end{cases}. \tag{12}$$

For each node, we use the mean value of the time lags of all the elements at one timestamp as the time lags of the corresponding node, which transforms the element-level time lags into node-level time lags. Specifically, $\sum_{j=1}^{n}\delta_i^j$ first records the sum of n time lags, then $\sum_{j=1}^{n}\frac{1}{n}\delta_i^j$ reflects the mean time lag of node i. The time lags between nodes k and h can be represented by the sum of multiple node-level time lags between them, i.e., $\sum_{i=k}^{h}\sum_{j=1}^{n}\frac{1}{n}\delta_i^j$. Since the time lag is inversely proportional to temporal dependencies, we use the reciprocal of the time lag between two nodes to reflect this relationship. This hypothesis also fit the properties of time series that the adjacent series have more significant influence than non-adjacent series. In this way, $\widetilde{A}$ is derived by applying the time lags between nodes k and h to the corresponding positions. According to the above derivation, $\widetilde{A}$ can simulate the decay connection between nodes due to the presence of missing values. With the derived $\widetilde{A}$, the GCN formula can distinctively aggregate neighbor information of nodes and encode a better latent code of incomplete nodes through the GCN encoder described in Section 4.2. Besides, the effectively inferred code will lead to a more precise reconstruction of original time series by GAN inversion.

4.4. Losses for training

In the pretraining stage, the purpose of G is to produce new sample $\widehat{X}$ that is most similar to X and fool D. We add MSE and adversarial loss to the loss of G. The reconstruction loss ensures that $\widehat{X}$ are consistent with X as follows:

$$L_G = \mathrm{MSE}(X, \widehat{X}) - \mathrm{D}(X, \widehat{X}). \tag{13}$$

Similar to G, E encodes X into a latent code to precisely reconstruct X:

$$L_E = \mathrm{MSE}(X, \widehat{X}). \tag{14}$$

D aims to distinguish $\widehat{X}$ from X and is trained to judge X as true and judge $\widehat{X}$ as false. To meet the requirements of the update process, the loss of D is formulated as below:

$$L_D = \mathrm{D}(X, \widehat{X}) = \mathrm{D}(\widehat{X}) - \mathrm{D}(X). \tag{15}$$

The model is trained to find an equilibrium between G and D with the joint optimization of the above losses. In the inversion stage, we utilize the hybrid method to implement GAN inversion where E, G and D are fixed through the inversion stage. The fixed E takes X^{inc} as input to infer a latent code to precisely reconstruct X. Then the latent code is fed into G to ensure that $\widehat{X}$ are consistent with X^{inc} in the observed components specified by M, which is different from that in the pretraining stage. The loss as follows is used to update the latent code by gradient decent:

$$L_I = \mathrm{MSE}\big(X^{inc}, \widehat{X} \odot M\big) = \mathrm{MSE}\big(X^{inc}, \mathrm{G}\big(\mathrm{E}\big(X^{inc}\big)\big) \odot M\big). \tag{16}$$

When the model converges after some epochs, $\widehat{X}$ will reach a credible level. We use $\widehat{X}$ to impute the missing values of X^{inc} in the corresponding positions according to M formulated as below:

$$X^{imp} = X^{inc} \odot M + \widehat{X} \odot (1 - M). \tag{17}$$

5. Experiments

5.1. Implementation details

In the experiments, we first evaluate the performance of our proposal against existing methods in terms of imputation error on various datasets in different missing rates. Then we evaluate the visualization effect from different angles by PCA [45], t-SNE [46], KDE [47] and Q-Q plot [48]. Afterwards, the converge performance of different GAN inversion methods is analysed and an ablation study for various components is conducted. All input values are scaled with min–max normalization. We test each imputation method at missing rates of 0.3, 0.5 and 0.7. To achieve a adversarial balance, the generator is trained 5 times while the discriminator is trained 1 times at each epoch. Before imputation, the generator is pretrained with complete datasets. After acquiring a well-trained generator, we fix its parameters and set the encoder or latent codes to be trainable as discussed in Section 4.2. When training for imputation, the deteriorated datasets are taken as input with missing masks. Since the original datasets contain no missing values, we simulate the presence of missing values by missing completely at random (MCAR) to form corresponding missing masks. The experiments are

implemented with an Intel i7 11700 k CPU, 32 GB memory and an NVIDIA RTX 3060ti GPU. The deep learning framework is PyTorch 1.7.1 and cu110.

We use six datasets in the experiments: Sine, Stock, Energy, Beijing Air-Quality Data (AQ), Letter and Spam. The samples of each dataset are 1000, 3685, 19735, 35064, 20000 and 4601, respectively. The dimensions of attributes are 10, 6, 26, 10, 16 and 57 respectively. The first Sine is a synthetic dataset and the Stock is a dataset of historical Google stocks. The left are from the UCI repository. The Sine dataset consists of 10 independent sine waves, where each wave provides continuous values and has random shifts in phase θ and frequency ω. For each wave i, the sine function is $x_i(t) = \sin(\omega t + \theta)$, where ω $\mathrm{Uniform}[0, 0.1]$ and θ $\mathrm{Uniform}[0, \pi]$. The intermediate dimensions of latent codes are set to be 4, 4, 10, 10, 5 and 10 for Sine, Stock, Energy, AQ, Letter and Spam, respectively. A more careful design of the dimensions for each dataset may improve the performance. Sine, Stock, Energy and AQ datasets are typical time series data while Letter and Spam are tabular data. We utilize these tabular data to test the adaptation of our proposal on different types of data. The epoch for pretraining is 1000 and batchsize is 300. The epoch for the optimization of latent codes is 10000. We use the Adam optimizer with learning rate 0.001 to update the model.

We compare our proposal with several commonly used imputation methods:

- **GAIN**: A generative adversarial imputation model with the hint mechanism.
- **GRIN**: A recent graph imputation model combined with RNN structure.
- **GINN**: A missing value imputer with a graph autoencoder and the adversarial training.
- **GRAPE**: it imputes features in a bipartite graph by viewing observations and feature as different nodes.
- **MICE**: It iteratively infers missing values by using multiple regression models and chained equations.

5.2. Imputation error

In the first experiment we quantitatively evaluate the imputation performance of our proposal by the root mean squared error (RMSE) as a performance metric. The result is calculated on the scaled imputed data due to the normalized input as discussed above. We vary the fraction of missing rate to further test the adaptation of the proposal to different scenarios. Table 1 summarizes the RMSE and standard deviations of tested imputation methods. As the table shows, our proposal exhibits competitive performance in most scenarios. With the increase of missing rates, our proposal deteriorated slightly in comparison with the notable performance decline of other methods. Moreover, in conditions of high missing rates and datasets with temporal features such as Stock and Energy, our proposal consistently outperforms other benchmarks. It can be attributed to the fact that our method models the temporal irregularity due to the presence of missing values, which exerts better aggregation of adjacent information in incomplete time series. Though GAIN and KNN outperform our method or show competitive results in some scenarios with tabular data, they fail to maintain the equal advantages on time series datasets. This is most likely due to the fact that they specialize in tabular data and do not account for data with irregular temporal features.

We use the Decision Tree classifier to conduct post-imputation classification tasks to further evaluate imputation quality. Due to the fact that some data contains no labels for classification, the KNN algorithm, one of the commonly used clustering methods [49], is introduced to achieve original clustering labels. For the tested datasets, the number of clusters is 15 (Sine), 5 (Stock), 20 (Energy), 15 (AQ), 30 (Letter) and 20 (Spam), respectively. The classifier is trained on original data with labels and then is used to test imputed data derived in 0.3 missing rate by different methods.The results are shown in Table 2. Consistent with Table 1, our method provides more accurate classification results. Recent method including GRIN and GRAPE also show comparable performance in some cases. The reason for the improvements is that GAN inversion introduced in our method makes the model possess rich properties of original data, which is beneficial to classification tasks.

Table 1
RMSE and standard deviations on different datasets. Bold values indicate the best performance.

Dataset	Missing Rate	Ours	GAIN	GRIN	GINN	GRAPE	MCIE
Sine	0.3	0.2243(0.01)	0.2721(0.02)	**0.2017(0.01)**	0.3179(0.002)	0.2701(0.02)	0.2252(0.02)
	0.5	**0.2579(0.02)**	0.3258(0.02)	0.2753(0.02)	0.3342(0.002)	0.2975(0.02)	0.2654(0.01)
	0.7	**0.2743(0.01)**	0.3417(0.02)	0.2903(0.01)	0.3402(0.02)	0.3296(0.02)	0.3157(0.01)
Stock	0.3	0.0463(0.001)	0.0652(0.02)	0.0513(0.001)	0.1681(0.002)	0.0743(0.02)	**0.0423(0.02)**
	0.5	**0.0532(0.001)**	0.0882(0.03)	0.0581(0.001)	0.1742(0.002)	0.0827(0.02)	0.0752(0.03)
	0.7	**0.0656(0.002)**	0.1605(0.03)	0.0746(0.002)	0.1595(0.002)	0.1422(0.03)	0.1461(0.03)
Energy	0.3	0.1042(0.002)	0.1233(0.03)	0.1141(0.002)	0.1477(0.001)	0.1334(0.03)	**0.0949(0.01)**
	0.5	**0.1089(0.002)**	0.1537(0.02)	0.1451(0.002)	0.1637(0.002)	0.1421(0.02)	0.1123(0.01)
	0.7	**0.1179(0.002)**	0.1794(0.02)	0.1374(0.002)	0.1477(0.002)	0.1563(0.02)	0.1443(0.02)
AQ	0.3	0.0973(0.002)	0.1257(0.002)	**0.0921(0.002)**	0.1431(0.002)	0.1346(0.002)	0.0968(0.002)
	0.5	**0.0949**(0.002)	0.1359(0.002)	0.1269(0.002)	0.1657(0.002)	0.1442(0.002)	0.1046(0.002)
	0.7	**0.1214**(0.002)	0.1483(0.002)	0.1324(0.002)	0.1789(0.002)	0.1582(0.002)	0.1363(0.002)
Letter	0.3	0.1087(0.01)	0.1161(0.01)	0.1422(0.01)	0.1398(0.002)	**0.1032(0.01)**	0.1502(0.001)
	0.5	0.1324(0.01)	**0.1207(0.02)**	0.1502(0.02)	0.1457(0.002)	0.1326(0.02)	0.1651(0.002)
	0.7	**0.1371(0.02)**	0.1546(0.01)	0.1678(0.01)	0.1528(0.002)	0.1476(0.02)	0.1854(0.003)
Spam	0.3	**0.0532(0.003)**	0.0567(0.002)	0.0731(0.001)	0.1572(0.002)	0.0627(0.002)	0.0732(0.002)
	0.5	0.0557(0.003)	**0.0548(0.002)**	0.0761(0.002)	0.1723(0.002)	0.0768(0.002)	0.0843(0.003)
	0.7	**0.0613(0.01)**	0.0855(0.03)	0.1185(0.02)	0.2187(0.002)	0.0817(0.03)	0.981(0.003)

Table 2
Classification accuracy on different datasets. Bold values indicate the best performance.

Datasets	Ours	GAIN	GRIN	GINN	GRAPE	MICE
Sine	**0.762**	0.722	0.738	0.671	0.718	0.681
Stock	**0.955**	0.953	0.947	0.925	0.946	0.937
Energy	0.721	0.671	**0.743**	0.675	0.689	0.708
AQ	**0.781**	0.739	0.762	0.694	0.723	0.744
Letter	0.663	**0.682**	0.634	0.659	0.674	0.651
Spam	0.672	**0.749**	0.652	0.673	0.681	0.714

5.3. Visualization

To analyse how well the imputed data X^{imp} derived in Section 4.4 resembles the original input X, we choose PCA, t-SNE, KDE and Q-Q plot to visualize the comparison results so as to give a qualitative assessment of the learned representations. GAIN is chosen as the baseline in the experiment. PCA and t-SNE are applied to show the 2-dimensional representations of tested datasets. Note that t-SNE is sensitive to numerical changes and it is difficult for models to generate data that is completely consistent with the original input especially in high missing rates, which is determined by the characteristics of the imputation task. In a high missing rate, the imputed and original data are more likely to be inconsistent with each other and the corresponding visualization images will be quite different. Therefore, we only compare the results in 0.3 missing rate. In fact, given a partial input, there is not a single and fixed complete output and multiple reasonable imputation results are expected. [24,25] yield multiple reasonable imputed shapes given partial scans, which means the original and imputed shapes are not always exactly identical. Similar to these methods, our proposal and GAIN also output diverse reasonable imputation results by t-SNE, as depicted in Fig. 5(a) and (b). This means multiple possible temporal patterns of time series are inferred conditioned on the observations. Besides, our method produces more authentic overlap with original time series in Fig. 4(a) and Fig. 5(a) than GAIN. For tabular data, both methods output similar visual images by PCA in Fig. 4(c) and (d), as well as t-SNE in Fig. 5(c) and (d). These results are consistent with Table 1, which demonstrate the effectiveness of our method for time series imputation.

To compare the distributions of both imputed and original datasets, we select one attribute from each dataset to visualize the results by KDE. In Fig. 6, we observe that imputed data by our method are almost perfectly in sync with original input, especially tested on time series in Fig. 6(a) and (b). In addition to the relatively smooth curves, our proposal also fits well at the fluctuating curve as depicted in Fig. 6(c). In the last visualization experiment, we show the Q-Q plot for comparison from another angel. Fig. 7 summarizes the results where each column shows one attribute from each dataset. Both tested methods perform well on tabular, while the fitted points derived by our proposal are closer to the target lines than other competitor in case of time series data. The comparison results are consistent with above experiments.

5.4. Convergence analysis

As discussed in Section 4.2, we choose the hybrid method to conduct GAN inversion which optimizes the latent code from the pretrained encoder. To show its superiority over other two inversion methods, i.e., optimization and encoder, we draw their convergence curves of RMSE in Fig. 8 on different datasets during training. The missing rate is 0.3. We can explicitly observe that all three inversion methods lead to stable convergences after several epochs. It is noteworthy the best RMSE emerges after trained with

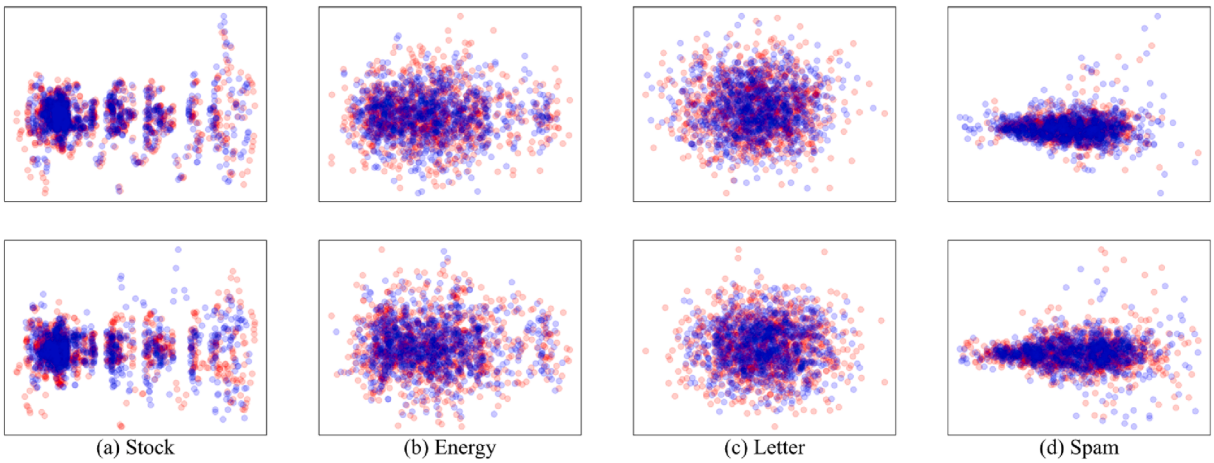

Fig. 4. PCA (red for real data and blue for imputed data) by our method (top row) and GAIN (bottom row), respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

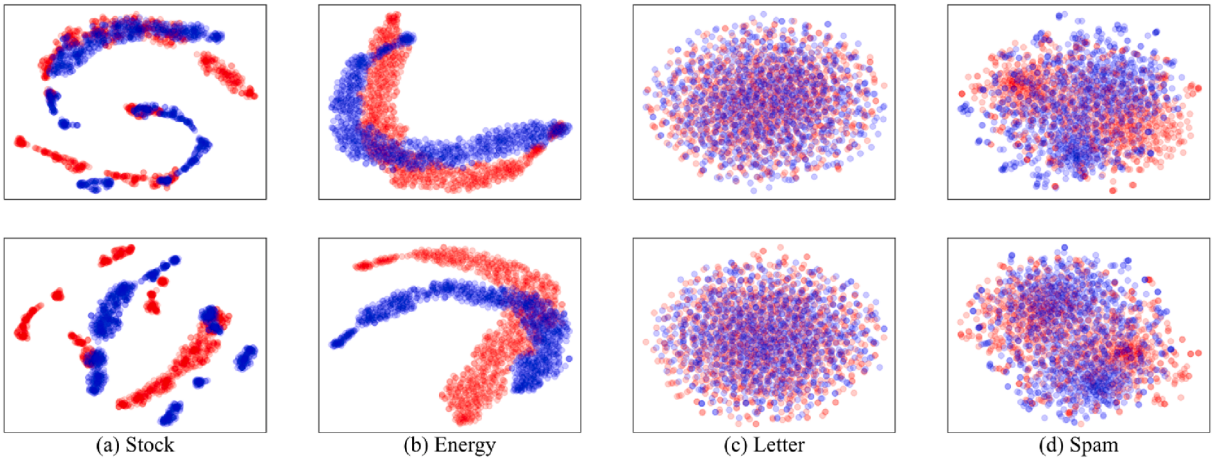

Fig. 5. t-SNE (red for real data and blue for imputed data) by our method (top row) and GAIN (bottom row), respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

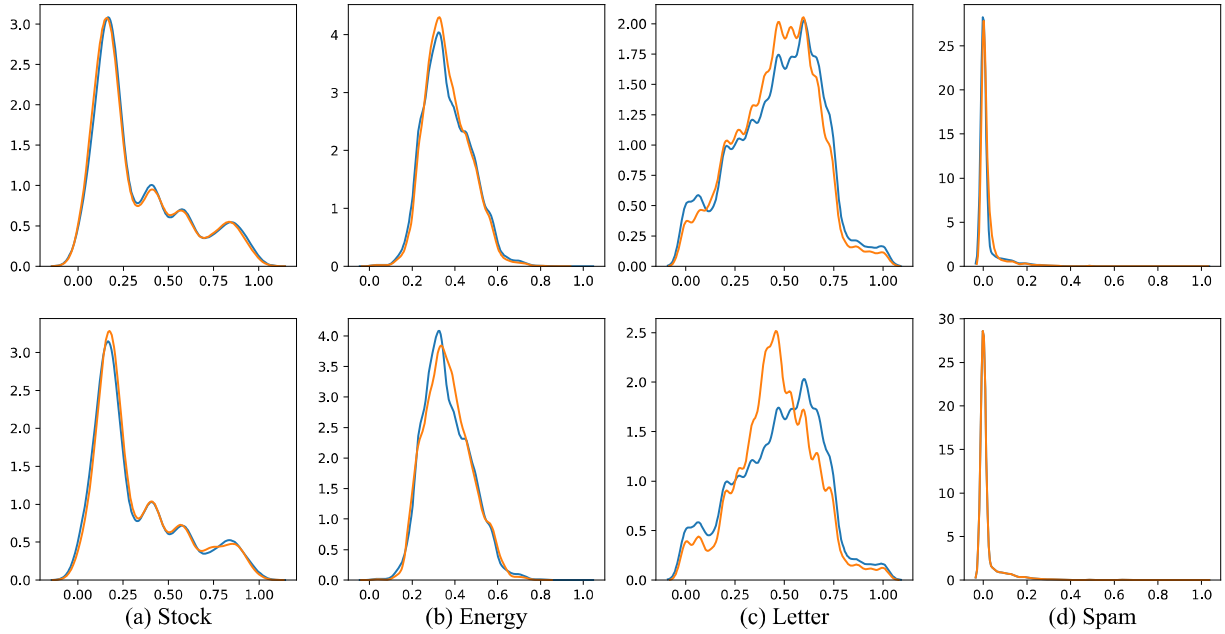

Fig. 6. KDE (blue for real data and orange for imputed data) by our method (top row) and GAIN (bottom row), respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

hybrid and optimization methods on each dataset. Compared with the optimization method which achieves similar performance at the end of training, the hybrid method converges faster when trained with the same epochs. Compared with the encoder method which shows faster convergence and shares the same initial state of latent codes, our hybrid method achieves lower RMSE when trained with the same epochs in most cases. This means our method can produce the best performance and faster convergence speed simultaneously.

5.5. Ablation study

In the section, we analyse the influence of different components, i.e., the GAN inversion and decay connection measured by RMSE in Table 3. The missing rate is set to be 0.5 and other implementation details follow the settings in Section 5.1. Ours w/o GI and Ours w/o DC indicate the method without GAN inversion and decay connection, respectively. As we can observe, every component does have a positive effect and a combination of both clearly leads to better imputation performance than separate utilization. Given datasets with temporal correlations, the GAN inversion and decay connection exert similar influence on imputation performance. We

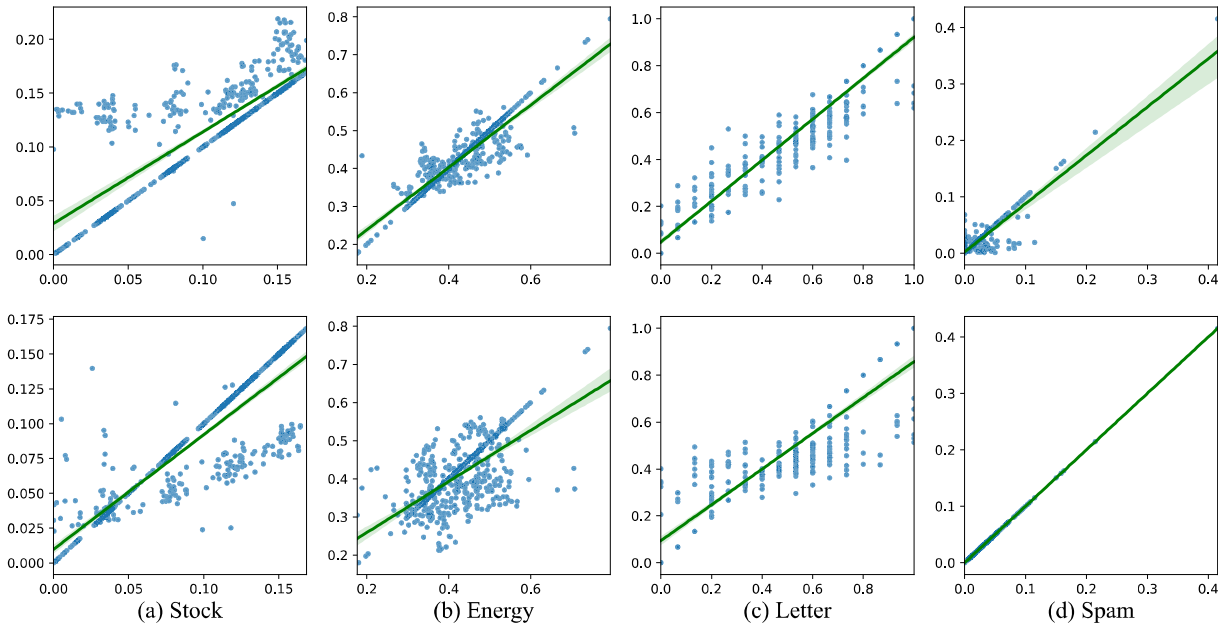

Fig. 7. QQ-plot (blue for fitted points and green for target lines) by our method (top row) and GAIN (bottom row), respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

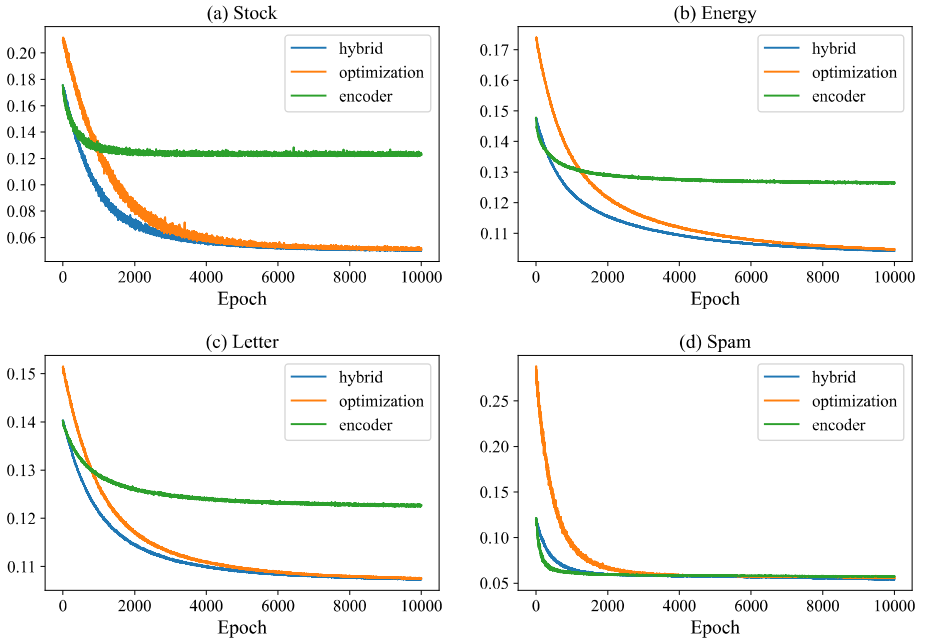

Fig. 8. Convergence curves of different inversion methods.

Table 3
Ablation results in terms of RMSE of each component.

Model	Energy	Letter
Ours	**0.1089(0.002)**	**0.1224(0.01)**
Ours w/o GI	0.1142(0.002)	0.1483(0.01)
Ours w/o DC	0.1181(0.002)	0.1367(0.01)

Bold values indicate the best performance.

can attribute the result to the reason that each component can be viewed as a method to obtain latent codes for reconstruction from a different perspective, one is from prior knowledge through hybrid inversion, the other is from irregular features of incomplete data. When tested on tabular data, the GAN inversion shows more contribution than decay connection. This is probably due to the fact that tabular data contain fewer temporal correlations than time series, which leads to less influence of the decay connection to get better latent codes while the GAN inversion can always encode appropriate latent codes from prior knowledge.

5.6. Limitations

Though our proposal achieves superior imputation performance evaluated from different angles, some limitations remain unsettled. The first point lies in that our proposal will construct a node for each sample, which inevitably exerts a huge graph in case of a large-scale dataset. This process of graph construction may consume more computing resources even than the whole training stage. The second point is the need for complete data in the pretraining stage. Actually we may have no access to the desired complete data, which will affect the practical applicability of the model in real scenarios.

6. Conclusion

We present a generative model for time series imputation with GAN inversion and decay connection. The novel model tries to characterize the irregular temporal dependencies of incomplete time series and incorporate the quantization into the GCN to better aggregate adjacent information and infer a better latent code. With the adoption of GAN inversion, the latent code in the latent space of a pretrained GAN will guarantee a faithful reconstruction. This is the first time that the GAN inversion is extended to the GCN so that it can address the unique features of time series imputation. Various quantitative and qualitative experiments show that our proposal achieves state-of-the-art or competitive imputation performance.

CRediT authorship contribution statement

Longfei Xu: Conceptualization, Methodology, Software. **Lingyu Xu:** Supervision. **Jie Yu:** Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

References

[1] X. Yi, Y. Zheng, J. Zhang, T. Li. ST-MVL: filling missing values in geo-sensory time series data, in: International Joint Conference on Artificial Intelligence, 2016.
[2] H. Tan, G. Feng, J. Feng, W. Wang, Y.J. Zhang, F. Li, A tensor-based method for missing traffic data completion, Transp. Res. Part C: Emerg. Technol. 28 (2013) 15–27.
[3] K.J.M. Janssen, A.R.T. Donder, H.F.E. Jr, Y. Vergouwe, Q. Chen, D.E. Grobbee, K.G. Moons, Missing covariate data in medical research: to impute is better than to ignore, J. Clin. Epidemiol. 63 (2010) 721–727.
[4] D.V. Mehrotra, F. Liu, T. Permutt, Missing data in clinical trials: control-based mean imputation and sensitivity analysis, Pharm. Stat. 16 (2017) 378–392.
[5] S. Zhang, Nearest neighbor selection for iteratively kNN imputation, J. Syst. Softw. 85 (2012) 2541–2552.
[6] P.J. García-Laencina, J.-L. Sancho-Gómez, A.R. Figueiras-Vidal, Pattern classification with missing data: a review, Neural Comput. Appl. 19 (2) (2010) 263–282.
[7] J.S. Murray, Multiple imputation: a review of practical and theoretical findings, Stat. Sci. 33 (2018) 142–159. Mice extension.
[8] J. Han, S. Kang, Active learning with missing values considering imputation uncertainty, Knowl.-Based Syst. 224 (2021), 107079.
[9] Y. Bengio, F. Gingras. Recurrent neural networks for missing or asynchronous data, in: Advances in Neural Information Processing Systems, 1995.
[10] J. Yoon, W.R. Zame, M. van der Schaar, Estimating missing data in temporal data streams using multi-directional recurrent neural networks, IEEE Trans. Biomed. Eng. 66 (5) (2019) 1477–1490.
[11] Z. Che, S. Purushotham, K. Cho, D. Sontag, Y. Liu, Recurrent neural networks for multivariate time series with missing values, Sci. Rep. 8 (2018) 1–12.
[12] W Cao, D. Wang, J. Li, H. Zhou, L. Li, Y. Li. Brits: Bidirectional recurrent imputation for time series, in: Advances in Neural Information Processing Systems, 2018.
[13] H. Khan, X. Wang, H. Liu, Handling missing data through deep convolutional neural network, Inf. Sci. 595 (2022) 278–293.
[14] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, Y. Bengio. Generative Adversarial Nets. in: Advances in Neural Information Processing Systems, 2014.
[15] Y. Luo, X. Cai, Y. Zhang, J. Xu. Multivariate time series imputation with generative adversarial networks. in: Advances in Neural Information Processing Systems, 2018.
[16] Y. Luo, Y. Zhang, X. Cai, X. Yuan. E2gan: End-to-end generative adversarial network for multivariate time series imputation, in: International Joint Conference on Artificial Intelligence, 2019.
[17] J. Yoon, J. Jordon, M. Schaar. Gain: Missing data imputation using generative adversarial nets, in: International Conference on Machine Learning, 2018.
[18] T.N. Kipf, M. Welling. Semi-supervised classification with graph convolutional networks. in: International Conference on Learning Representations, 2016.
[19] I. Spinelli, S. Scardapane, A. Uncini, Missing data imputation with adversarially-trained graph convolutional networks, Neural Netw. 129 (2020) 249–260.
[20] H. Taguchi, X. Liu, T. Murata, Graph convolutional networks for graphs containing missing features, Futur. Gener. Comput. Syst. 117 (2021) 155–168.
[21] A. Cini, I. Marisca, C. Alippi. Filling the g_ap_s: Multivariate time series imputation by graph neural networks. arXiv preprint arXiv:2108.00298, 2021.
[22] A. Creswell, A.A. Bharath, Inverting the generator of a generative adversarial network, IEEE Trans. Neural Networks Learn. Syst. 30 (7) (2019) 1967–1974.
[23] J. Zhu, Y. Shen, D. Zhao, B. Zhou. In-domain gan inversion for real image editing, in: European Conference on Computer Vision, 2020.

[24] J. Zhang, X. Chen, Z. Cai, L. Pan, H. Zhao, S. Yi, C.C. Loy. Unsupervised 3d shape completion through gan inversion, in: IEEE Conference on Computer Vision and Pattern Recognition, 2021.
[25] H. Arora, S. Mishra, S. Peng, K. Li, A. Mahdavi-Amiri. Multimodal Shape Completion via Implicit Maximum Likelihood Estimation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2022.
[26] W. Wang, L. Niu, J. Zhang, X. Yang, L. Zhang. Dual-path Image Inpainting with Auxiliary GAN Inversion, in: IEEE Conference on Computer Vision and Pattern Recognition, 2022.
[27] Y. Zhang, B. Zhou, X. Cai, W. Guo, X. Ding, X. Yuan, Missing value imputation in multivariate time series with end-to-end generative adversarial networks, Inf. Sci. 551 (2021) 67–82.
[28] Y. Wang, D. Li, X. Li, M. Yang, PC-GAIN: pseudo-label conditional generative adversarial imputation networks for incomplete data, Neural Netw. 141 (2021) 395–403.
[29] X. Miao, Y. Wu, J. Wang, Y. Gao, X. Mao, J. Yin. Generative semi-supervised learning for multivariate time series imputation, in: AAAI conference on artificial intelligence, 2021.
[30] S. Yang, M. Dong, Y. Wang, C. Xu, Adversarial recurrent time series imputation, IEEE Trans. Neural Networks Learn. Syst. (2020) 1–12.
[31] J. Ho, A. Jain, P. Abbeel. Denoising diffusion probabilistic models. in: Advances in Neural Information Processing Systems, 2020.
[32] Y. Tashiro, J. Song, Y. Song. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. in: Advances in Neural Information Processing Systems, 2021.
[33] J.M.L. Alcaraz, N. Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. arXiv preprint arXiv:2208.09399, 2022.
[34] J. You, X. Ma, Y. Ding. Handling missing data with graph representation learning. in: Advances in Neural Information Processing Systems, 2020.
[35] C. Zhu, X. Ma, C. Zhang, W. Ding, J. Zhan, Information granules-based long-term forecasting of time series via BPNN under three-way decision framework, Inf. Sci. 634 (2023) 696–715.
[36] W. Xia, Y. Zhang, Y. Yang, J.H. Xue, B. Zhou, M.H. Yang, Gan inversion: a survey, IEEE Trans. Pattern Anal. Mach. Intell. (2022).
[37] T. Wang, Y. Zhang, Y. Fan, J. Wang, Q. Chen. High-fidelity gan inversion for image attribute editing, in: IEEE Conference on Computer Vision and Pattern Recognition, 2022.
[38] T. Kang. Multiple GAN Inversion for Exemplar-based Image-to-Image Translation, in: IEEE International Conference on Computer Vision, 2021.
[39] Z.C. Lipton, S. Tripathi. Precise recovery of latent vectors from generative adversarial networks. arXiv preprint arXiv:1702.04782, 2017.
[40] D. Bau, J.Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, A. Torralba, Inverting layers of a large generator. International Conference on Learning Representations, 2019.
[41] J. Luo, Y. Xu, C. Tang, J. Lv. Learning inverse mapping by autoencoder based generative adversarial nets, in: Advances in Neural Information Processing, 2017.
[42] D. Bau, J.Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, A. Torralba. Seeing what a gan cannot generate, in: IEEE International Conference on Computer Vision, 2019.
[43] X. Pan, X. Zhan, B. Dai, D. Lin, C.C. Loy, P. Luo, Exploiting deep generative prior for versatile image restoration and manipulation, IEEE Trans. Pattern Anal. Mach. Intell. 44 (2022) 7474–7489.
[44] T. Wei, X. Li, V. Stojanovic, Input-to-state stability of impulsive reaction-diffusion neural networks with infinite distributed delays, Nonlinear Dyn. 103 (2021) 1733–1755.
[45] F.B. Bryant, P.R. Yarnold. Principal-components analysis and exploratory and confirmatory factor analysis, Reading and understanding multivariate statistics. (1995) 99–136.
[46] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.
[47] Z.I. Botev, J.F. Grotowski, D.P. Kroese, Kernel density estimation via diffusion, Ann. Stat. 38 (2010) 2916–2957.
[48] J.I. Marden, Positions and QQ plots, Stat. Sci. 19 (2004) 606–614.
[49] R. Zhang, X. Ma, J. Zhan, Y. Yao, 3WC-D: a feature distribution-based adaptive three-way clustering method, Appl. Intell. (2022) 1–19.