Contents lists available at ScienceDirect

Neurocomputing

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

A data imputation method for multivariate time series based on generative adversarial network

Zijian Guo[a,b], Yiming Wan[c,d], Hao Ye[a,b,*]

[a] Department of Automation, Tsinghua University, Beijing 100084, PR China
[b] Beijing National Research Center for Information Science and Technology (BNRist), Beijing 100084, PR China
[c] School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, PR China
[d] Key Laboratory of Image Processing and Intelligent Control, Huazhong University of Science and Technology, Ministry of Education, PR China

ARTICLE INFO

Article history:
Received 17 August 2018
Revised 3 February 2019
Accepted 3 June 2019
Available online 11 June 2019

Communicated by Prof. Zidong Wang

Keywords:
Multivariate time series
Missing data
Imputation
Distribution
Generative adversarial networks

ABSTRACT

Multivariate time series (MTS) processing plays an important role in many fields such as industry, finance, medical, etc. However, the presence of missing data in MTS makes data analysis process more complicated. To address this issue, MTS imputation reconstructs missing data with a high accuracy by exploiting a precise model of MTS distribution. Despite being an effective tool for modeling distribution on images, generative adversarial networks (GANs) have limitations in modeling MTS distribution. In this paper, to improve MTS imputation performance, a multivariate time series generative adversarial network (MTS-GAN) is proposed for MTS distribution modeling by introducing the multi-channel convolution into GANs. It is then applied to MTS imputation by formulating a constrained MTS generation task. Experimental results show that MTS-GAN performs well in modeling MTS distribution. Compared with several approaches, the proposed MTS-GAN based imputation method not only achieves a higher imputation accuracy under different missing rates, but also performs more robustly as the missing rate increases.

1. Introduction

Multivariate time series (MTS) recording time-varying values of multiple variables are able to reflect not only the time-varying trend of each variable, but also the coupling relationships among variables, which makes the analysis of them play an important role in industrial fields like process monitoring [1], fault diagnosis [2] and some other fields such as finance [3], meteorology [4], physics [5] and medical [6]. However, the unavoidable missing data caused by disturbances or even malfunction in the data sampling and transmission procedures may bring great difficulties to MTS analysis. Missing data usually bring three types of problems: (i) loss of efficiency, (ii) more complicated data analysis process, and (iii) serious bias owing to the differences between complete and missing data [7,8]. According to Barnard and Meng [7], there is little one can do about problem (i) after the data collection process.

Deletion and imputation methods are two important kinds of approaches to handle the problems of missing data. Deletion methods include listwise deletion and pairwise deletion [9]. Listwise deletion deletes all cases containing missing data and only reserves the cases with complete data, while in pairwise deletion, cases are excluded only when they have missing data on the variables involved in some operations [10]. In spite of its simplicity, deletion methods may lose much useful information when the missing ratio is large. Imputation methods, unlike deletion methods, aim to make full use of available information under observations and impute missing data to obtain a complete dataset [11–14].

Imputation is helpful to deal with problem (ii) and (iii). After imputation, a complete dataset can be obtained, and then many standard complete-data methods can be readily applied. Therefore, problem (ii) can be avoided [7]. Besides, a good imputation method is able to reconstruct missing data with a high accuracy by exploiting a precise model of data distribution, which can reduce bias between complete and missing data, and thus contribute to reducing the influence of problem (iii).

Nowadays, many researchers have been involved in studying the field of imputation, and a large number of data imputation methods have been proposed. Similar to [8], we divide these methods into three types: (i) simple data driven, (ii) model based, and (iii) deep learning based. Simple data driven imputation methods include hot deck [15], cold deck [16], mean/median/mode imputation [17] and so on. Mean/median/mode imputation replaces the missing values with the mean/median/mode value of a variable, but may introduce bias and change the statistical characteristics of original data such as variance and covariance [17]. Hot deck and cold deck imputation are usually used in questionnaire surveys and

* Corresponding author.
E-mail addresses: guozj15@mails.tsinghua.edu.cn (Z. Guo), ywan@hust.edu.cn (Y. Wan), haoye@mail.tsinghua.edu.cn (H. Ye).

https://doi.org/10.1016/j.neucom.2019.06.007
0925-2312/© 2019 Elsevier B.V. All rights reserved.

often make continuous data collapse into categories, thus cannot be used for MTS [9].

The typical model based imputation methods include interpolation based [18,19], singular value decomposition (SVD) based [11], regression based [12,20–22], likelihood based [23] and K-nearest neighbor (KNN) based [13,24,25] methods. Interpolation based techniques (e.g. cubic interpolation, spline interpolation) estimate the missing data by using their immediately preceding and succeeding values, which makes themselves tend to fail when many successive values are missing in a variable of an MTS [26]. SVD based imputation methods assume that the variables are linearly correlated, and therefore may give a poor performance for a nonlinear MTS where the assumption is violated [26]. Regression based imputation methods, such as linear regression, support vector machines and multi-layer perceptrons (MLPs), aim to use a set of observed variables to predict the missing data in another set of variables. Therefore, they are not applicable to cases where missing data exists in all variables. Likelihood based imputation methods describe data as a parameterized-model, and then use different kinds of EM (Expectation-Maximization) algorithms to estimate the parameters based on maximum likelihood or maximum a posteriori procedure [8]. However, the EM algorithms used in these methods are very sensitive to the initial values and often have a slow convergence rate [27]. Additionally, likelihood based imputation methods are only applicable to impute missing values for discrete attributes [8] whose domain is comprised of finite values such as gender, and therefore cannot be used for MTS analysis. The K-nearest neighbor (KNN) based imputation methods take advantage of some similarity measures (e.g. Euclidean distance, Mahalanobis distance) to find several samples from the dataset most similar to the given incomplete sample (i.e. the sample containing missing data) and use their values to impute missing values in the target sample [13,24,25]. The major drawback of this kind of methods is that in the imputation stage, for any incomplete sample, they have to search the whole dataset to find samples similar to the given one [28], which is computationally infeasible when handling a very large dataset. Moreover, the imputed values with the KNN based method are actually existing values in the dataset, rather than the constructed values [29].

In the past few years, various deep learning based methods have been proposed for missing data imputation, among which the recurrent neural networks (RNNs) and denoising autoencoders (DAEs) are the most popular ones. References [30,31] put the imputation into the training stage of RNN. They first initialize the missing values to be imputed and then, update them with the training of RNN. Since the imputation in [30,31] are accomplished within a supervised learning task for solving classification problems, they require labeled samples and cannot be applied to MTS without label information, which is often the case in real applications due to the unknown operating point change, disturbance, abnormality or faults of a process. As a kind of unsupervised deep learning methods, DAEs aiming to reconstruct the clean input from its corrupted version through a neural network as faithfully as possible, are also applied to missing data imputation [14,32]. Requiring no labels of data and being able to learn robust features make this kind of methods a good choice for MTS imputation.

In this paper, we aim at reducing the bias of MTS imputation, i.e. improving the performance of MTS imputation. To achieve this goal, it is essential to build a precise model of MTS distribution. Compared with DAEs which are good at finding a low-dimensional representation of the data, generative adversarial networks (GANs) [33] may give us a better choice in modeling data distribution. As a class of generative models, GANs specialize in modeling data distribution. More specifically, by training with original data, GAN can capture the distribution of original data [33–36] by making the distribution of its outputs (a group of synthetic data) approximate original data distribution.

However, although GANs have been used extensively in modeling distribution on images, such as image generation [37,38], image inpainting [39,40], style transfer [41,42] and so on, the application of GANs in modeling MTS is still limited. To the best of our knowledge, there are only two kinds of GANs having been used to model MTS distribution [43,44]. Reference [43] utilizes deep convolutional generative adversarial networks (DC-GAN) [38] to detect anomalous behavior of mechanical devices. Combining RNNs into GANs, reference [44] proposes recurrent generative adversarial networks (R-GAN) to produce realistic MTS, with an emphasis on their application to medical data. Note that the above two GANs have not yet been applied to the MTS imputation, and they have limitations in coping with MTS. The limitation of R-GAN is attributed to back-propagation through time (BPTT) training algorithm, which propagates the error backward over the entire input sequence in every time step. Therefore, R-GAN is inefficient in dealing with very long time series and has the difficulty in learning long-term dependencies [45]. As for DC-GAN, it is not suitable for capturing the distribution of MTS because the traditional convolutional neural networks (CNNs) [46] used in DC-GAN specialize in extracting features from two-dimensional (2-D) shapes which have the strong spatial regularities, such as images, whereas MTS do not have such characteristics. To improve the performance of CNN in processing MTS, Zheng et al. [47] develops a multi-channel convolutional neural network (MC-CNN) architecture. It first extracts features of each univariate time series of MTS, and then uses a fully connected MLP to learn the coupling relationships among the variables.

The limitations of the existing MTS imputation methods, the fact that GANs have a better potential in modeling data distribution, and the demerits of DC-GAN and R-GAN when they are applied to MTS distribution modeling, motivate us to propose a new variant of GANs for MTS distribution modeling, and develop an imputation method based on this variant to achieve a better MTS imputation performance. To the best of our knowledge, no report has been published in literature yet regarding GANs based MTS imputation. To deal with the limitations of DC-GAN in handling MTS mentioned above, we introduce the idea of MC-CNN [47] into DC-GAN, and propose the multivariate time series generative adversarial network (MTS-GAN) as well as a new MTS imputation method based on it. In simulation and experimental tests, the MTS-GAN performs well in modeling MTS distribution. Compared with the other competitive approaches, MTS-GAN based imputation method not only achieves a higher accuracy under different missing rates, but also performs more robustly as the missing rate increases.

The paper is organized as follows. Some preliminaries are provided in Section 2. In Section 3, to model MTS distribution precisely, we show the proposed MTS-GAN architecture and evaluate the performance of MTS-GAN in modeling MTS distribution in three common ways based on two toy datasets. In Section 4, regarding imputation as a constrained MTS generation task, we show an MTS-GAN based imputation method and the experimental results on MTS imputation based on three simulated datasets and a real-world dataset composed of historical field data collected from a high-speed railway. Section 5 concludes the paper.

2. Preliminaries

In this section, we first make a brief introduction to CNN and MC-CNN. Then GAN and a kind of its variants, DC-GAN, will be mentioned.

2.1. A brief introduction to CNN and MC-CNN

Specialized in extracting features from images, CNN [46] is mainly composed of convolutional layers and pooling

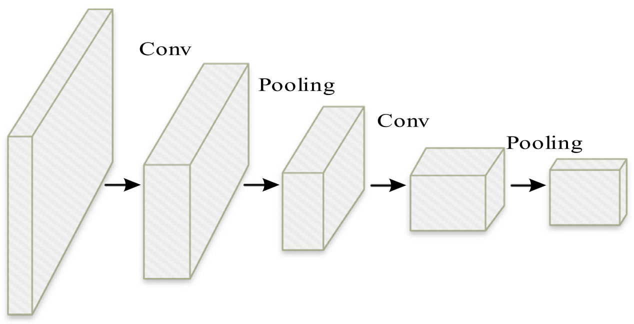

Fig. 1. The architecture of a two-layer CNN.

(subsampling) layers. Fig. 1 shows the architecture of a two-layer CNN, where the convolutional layers perform convolution with a predefined stride by sliding a 2-D kernel over the input, and the pooling layers perform subsampling process to reduce the number of network parameters.

Since MTS do not have the spatial regularities as images, to extract features from MTS better, MC-CNN is proposed in [47] whose architecture is shown in Fig. 2, where “Conv” denotes a convolutional layer. MC-CNN first extracts the features of each variable of the MTS in a single channel separately by performing convolutions and pooling alternatively, and then learns the coupling relationships among the multiple channels by using an MLP. To process univariate time series, the convolution in each channel uses 1-D kernel with stride 1. In the “Concatenate” block, the extracted features in each channel are flattened to a vector, and then N vectors are concatenated to a long vector. Note that the convolutional layers in Fig. 2 may have different kernel parameters.

2.2. A brief introduction to GAN and DC-GAN

Fig. 3 shows a GAN architecture, which is composed of a generator and a discriminator [33]. The generator G receives the latent random vector z sampled from a normal distribution or uniform distribution p_z and outputs a synthetic sample $G(z)$. The discriminator D takes either a training sample x or a synthetic sample $G(z)$ as input, and outputs a scalar indicating the probability that x or $G(z)$ follows the original data distribution p_{data}.

In the training stage, the generator tries to fool the discriminator as much as possible by solving the following optimization problem

$$\max_G E_{z\sim p_z(z)}[logD(G(z))] \tag{1}$$

and in the meantime, the discriminator tries to distinguish between original samples and the synthetic samples by

$$\max_D E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1-D(G(z)))] \tag{2}$$

After alternating the training of G and D, GAN can capture the distribution of the training data [33–36] by making the distribution of its outputs (a group of synthetic data) approximate original data distribution.

Unlike a traditional GAN (i.e. vanilla GAN) whose generator and discriminator are both composed of MLPs, DC-GAN [38] adopts the all convolutional net [48] in both the generator and the discriminator. Compared with the traditional CNN, the all convolutional net [48] replaces the pooling functions with strided convolutions (i.e. convolutions with stride $\geqslant$ 2). Fig. 4 shows an example of DC-GAN, in which “Conv Stride: 2” and “Deconv Stride: 2” represent the strided convolutions with stride 2 and the fractional-strided convolution (also called deconvolution in some papers) with stride 2, respectively. Since DC-GAN is proposed for handling images, all of the convolutional layers use 2-D kernels. DC-GAN can still be trained through solving the optimization problems represented by (1) and (2) alternatively. As pointed out by [38], DC-GAN is more stable than vanilla GAN.

3. MTS-GAN

A precise model of MTS distribution is essential for improving MTS imputation performance. Due to its strong abilities in modeling data distribution, GAN is a promising candidate for data imputation. Although the stability of GAN is further improved by DC-GAN [38], the convolutional architectures in DC-GAN have limitations in modeling MTS distribution, since MTS does not have the spatial regularities as images. In this section, to model MTS distribution precisely, we propose a modified version of GAN, called MTS-GAN. It introduces the idea of multi-channel convolutions of MC-CNN [47] into DC-GAN, which improves the performance in MTS distribution modeling. Some other modifications are also made, which will be shown in detail later.

3.1. Notations

Throughout the paper, we use $*(n, t)$ to denote the element of the n-th row and the t-th column of a matrix $*$. We denote an MTS dataset containing missing data as $\mathcal{X} = \{\mathbf{X}_m\}_{m=1}^{M}$, where $\mathbf{X}_m \in \mathbb{R}^{N\times T}$ is its m-th sample with N variables and length T in time, and M is the number of samples in $\mathcal{X}$. In addition, we use the indicator matrix δ_m where each elements $\delta_m(n,t)$ is a binary value to indicate whether $\mathbf{X}_m(n,t)$ is missing or not, i.e.

$$\delta_m(n,t) = \begin{cases} 1, & \text{if } \mathbf{X}_m(n,t) \text{ is not missing} \\ 0, & \text{if } \mathbf{X}_m(n,t) \text{ is missing} \end{cases} \tag{3}$$

$\mathcal{X}$ can be divided into the complete subset $\mathcal{X}^{com} = \{\mathbf{X}_m^{com}\}_{m=1}^{M_1}$ in which there is no missing data in all samples, and the incomplete subset $\mathcal{X}^{inc} = \{\mathbf{X}_m^{inc}\}_{m=1}^{M_2}$ in which there exists missing data in each sample. Obviously, for each $\mathbf{X}_m^{com}$, there is $sum(\delta_m^{com}) = NT$, and for each $\mathbf{X}_m^{inc}$, there is $sum(\delta_m^{inc}) < NT$, where $sum(\cdot)$ represents the sum of all the elements in the matrix.

3.2. MTS-GAN architecture

In this subsection, MTS-GAN is proposed mainly by introducing the multi-channel idea of MC-CNN into DC-GAN [38], and the architecture of its discriminator and generator are shown in Fig. 5 and Fig. 6 respectively. In the discriminator, the features of each variable of the MTS is extracted in a single channel by using multi-layer 1-D kernels based convolution with stride 2, and then the coupling relationships among the channels is modeled by an MLP, which outputs a scalar finally. In the discriminator, we use the same activations as DC-GAN [38]. More specifically, all layers adopt the leaky rectified linear units (LeakyReLU) activation [49], except for the output layer which uses sigmoid activation.

The generator shown in Fig. 6 is an inverse of the discriminator in Fig. 5. It takes d-dimensional latent random vector $z\sim U([0, 1]^d)$ following the uniform distribution as the input and performs fractional-strided convolutions (deconvolutions) within each single channel separately. Finally, each channel outputs one of the dimensions of the MTS. In the generator, the activations used are identical with DC-GAN [38]. Precisely, we use rectified linear units (ReLU) activation [50] for all layers in the generator, except Tanh function for the output layer.

Compared with DC-GAN which adopts 2-D convolutions, MTS-GAN performs 1-D kernel based strided convolutions or 1-D kernel based fractional-strided convolutions in each single channel to extract better features from MTS. Besides, inspired by MC-CNN [47],

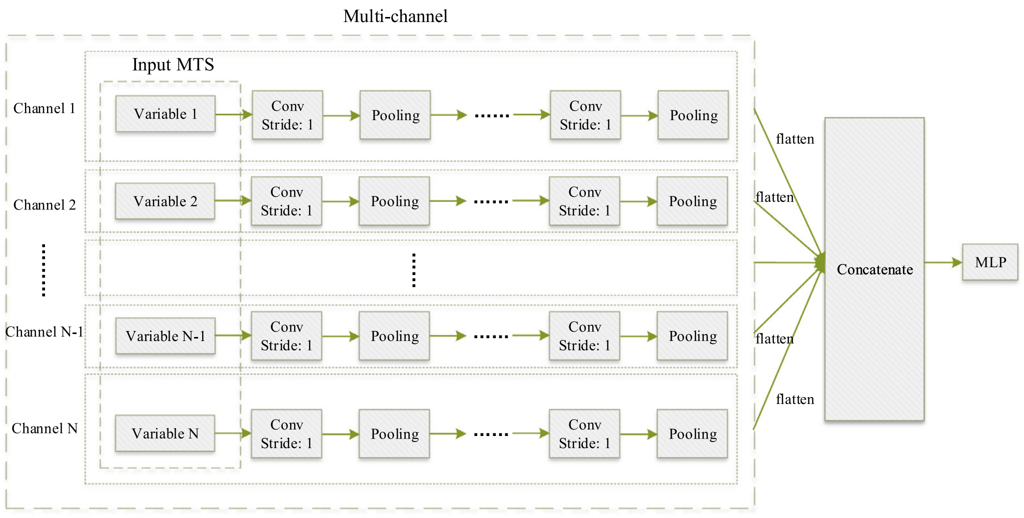

Fig. 2. An MC-CNN architecture for N-dimensional MTS.

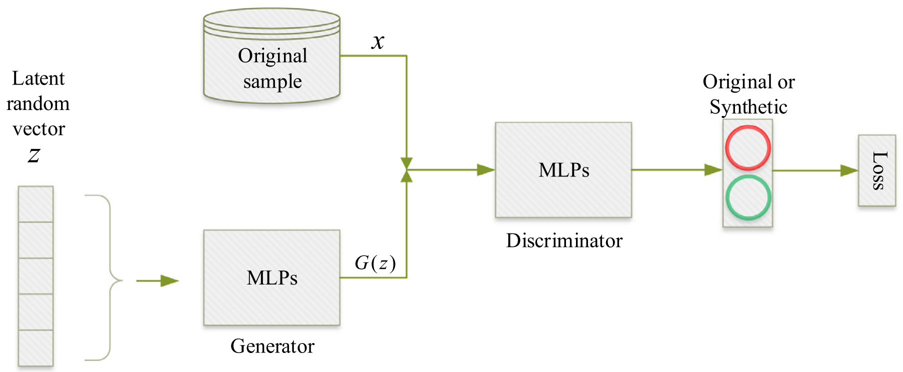

Fig. 3. A GAN architecture.

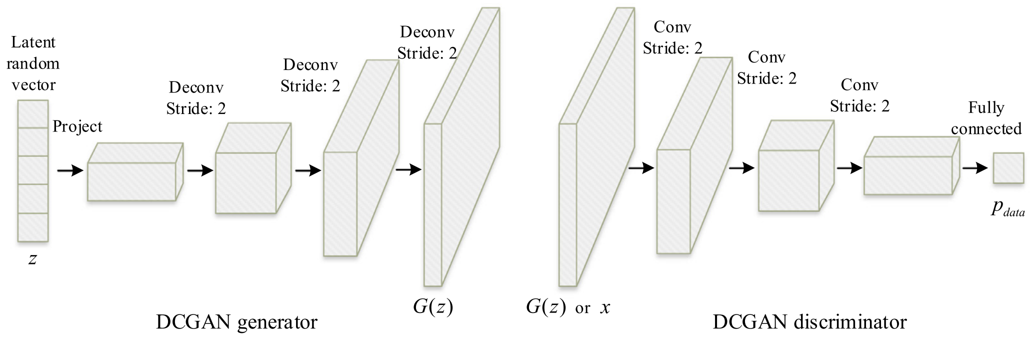

Fig. 4. A DC-GAN generator and discriminator, which contains three fractional-strided convolutional layers and three strided convolutional layers respectively.

MTS-GAN adds fully connected hidden layers into both generator and discriminator to learn the coupling relationships among all channels.

Identical with vanilla GAN and DC-GAN, the training objectives of the MTS-GAN generator and discriminator still follow (1) and (2) respectively, and the training algorithm of MTS-GAN is also the same as vanilla GAN, which can be found in [33].

3.3. *Experiments on evaluating the performance of MTS-GAN in modeling MTS distribution*

Before applying MTS-GAN to imputation, we need to evaluate the performance of MTS-GAN in modeling MTS distribution on a complete dataset and make sure that it is capable of capturing MTS distribution. Despite that a number of different measures have been proposed [38,51–53], no consensus has been achieved on which measure can best evaluate the performance of GAN so far [54]. In this subsection, we evaluate MTS-GAN in three common ways, (i) inspecting the quality of the synthetic samples visually, (ii) measuring the distance between the distribution of original data and that of the synthetic data based on Maximum Mean Discrepancy (MMD) [52], and (iii) using the interpolation based method [38] to evaluate whether the generative model overfits to training data. It is worth noting that all of the evaluation methods used in this subsection are based on a complete original dataset $\mathcal{X}^{com} = \{\mathbf{X}_m^{com}\}_{m=1}^{M_1}$ and a synthetic

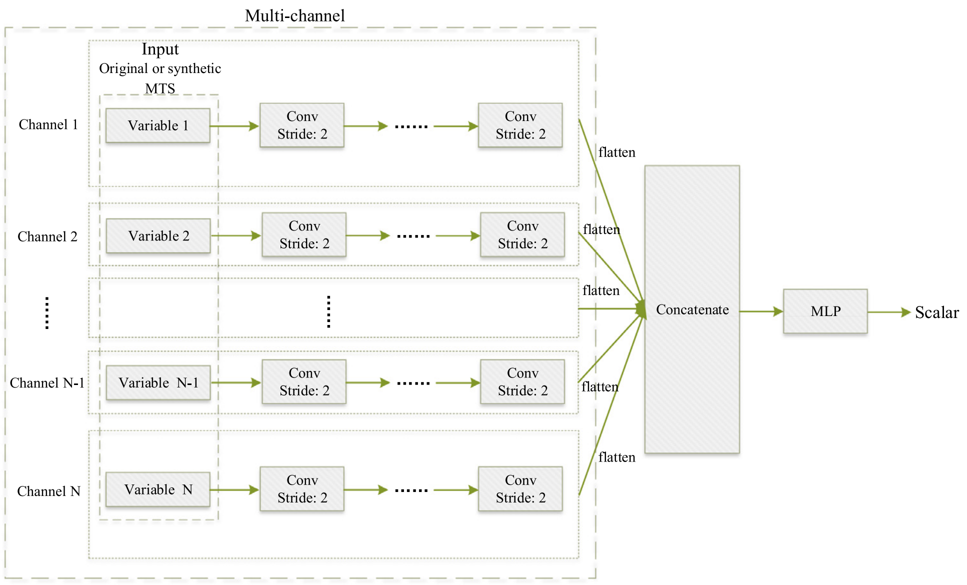

Fig. 5. The discriminator of the MTS-GAN for N-dimensional MTS.

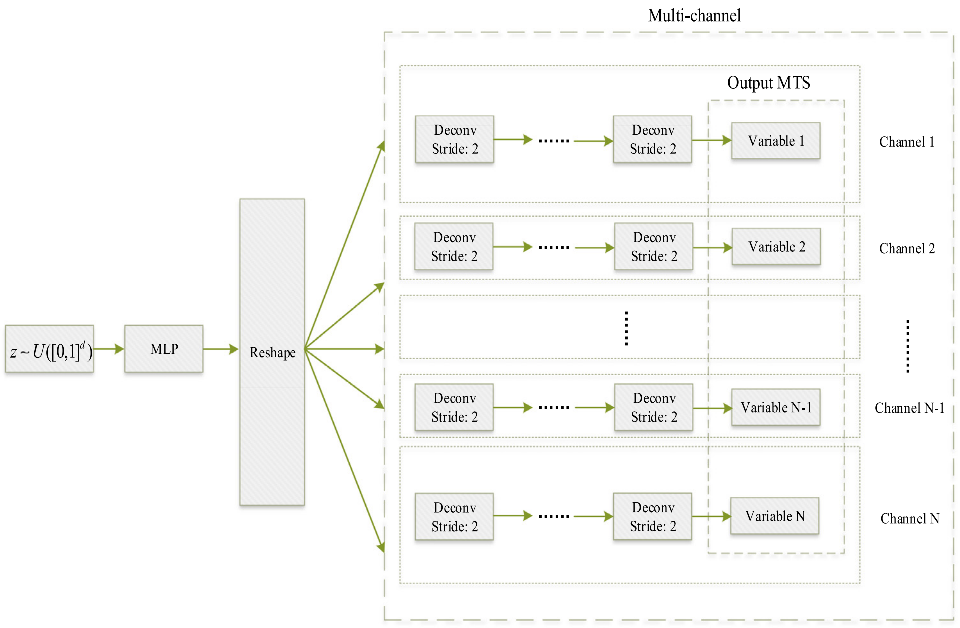

Fig. 6. The generator of the MTS-GAN for N-dimensional MTS.

dataset $\mathcal{X}^{gen} = \{\mathbf{X}_m^{gen}\}_{m=1}^{M_3}$ generated by an MTS-GAN trained with $\mathcal{X}^{com}$.

In [44], two datasets based on univariate sine waves and univariate Gaussian processes are used to evaluate GAN. Here, we extend them to three-dimensional nonlinear MTS.

1) Toy dataset1: suppose that three-dimensional multivariate sine waves are generated by

$$\left[Asin(2\pi ft+\phi)\;\; Asin(4\pi ft+2\phi)\;\; Acos(2\pi ft+\phi)\right]^T$$

where $[\cdot]^T$ denotes the transpose of a matrix. Amplitudes A, frequencies f, and phases ϕ for different samples are randomly generated and belong to [0.1,0.9], [1.0,5.0], $[-\pi, \pi]$, respectively, and keep constants in one sample. The length of each sample is 30. It is obviously that the coupling relationships among three dimensions are nonlinear.

2) Toy dataset2: suppose that three-dimensional MTS are generated by

$$\left[GP, abs(GP), 2\times GP\right]^T$$

where GP (called smooth functions in [44]) is a univariate signal sampled from a zero-mean Gaussian process whose covariance function are given by a RBF kernel, $abs(GP)$ denote its absolute value. The length of each sample is 30, too. Unlike the multivariate sine waves which are easy to be reproduced by GAN, MTS based on the smooth functions do not follow simple dynamics [44].

Based on the two toy datasets, which are first normalized to $[-1, 1]$, the proposed MTS-GAN is trained as described in Section 3.2 and further used to generate synthetic samples. Then the three methods mentioned above are adopted to evaluate the performance of MTS-GAN in modeling MTS distribution.

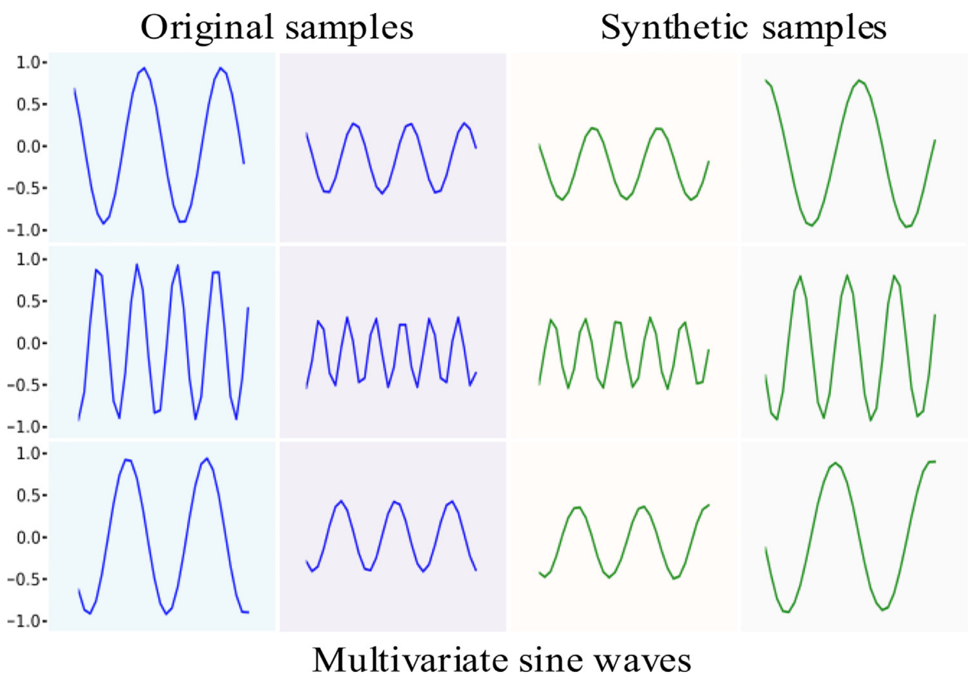

Fig. 7. Examples of the original and synthetic samples based on toy dataset1.

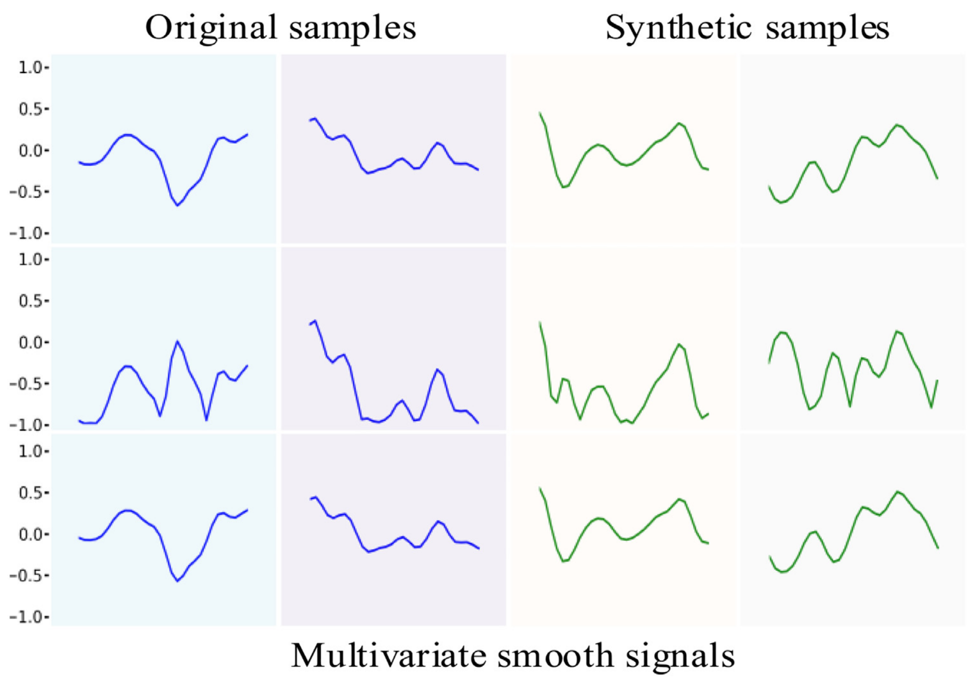

Fig. 8. Examples of the original and synthetic samples based on toy dataset2.

3.3.1. *Evaluating the quality of synthetic samples visually*

We first evaluate MTS-GAN by inspecting the quality of synthetic samples visually, which is often considered as the common and most intuitive way to evaluate GANs [54]. Figs. 7 and 8 show two examples of the original and synthetic samples based on the two toy datasets, respectively, in which, the left two blue columns are two 3-dimensional samples selected from the original complete dataset, and the right two green columns are the 3-dimensional synthetic samples. As can be seen from the figures, it is hard to distinguish between the original and the synthetic samples visually.

3.3.2. *Maximum Mean Discrepancy (MMD) based method [52]*

For some complex real-valued sequential data, it may be hard to visually evaluate the synthetic samples [44]. As a quantifiable measure, MMD evaluates GANs by measuring the distance between the distribution of original data and that of the synthetic samples. Given an original dataset $\mathcal{X}^{com}$ and a synthetic dataset

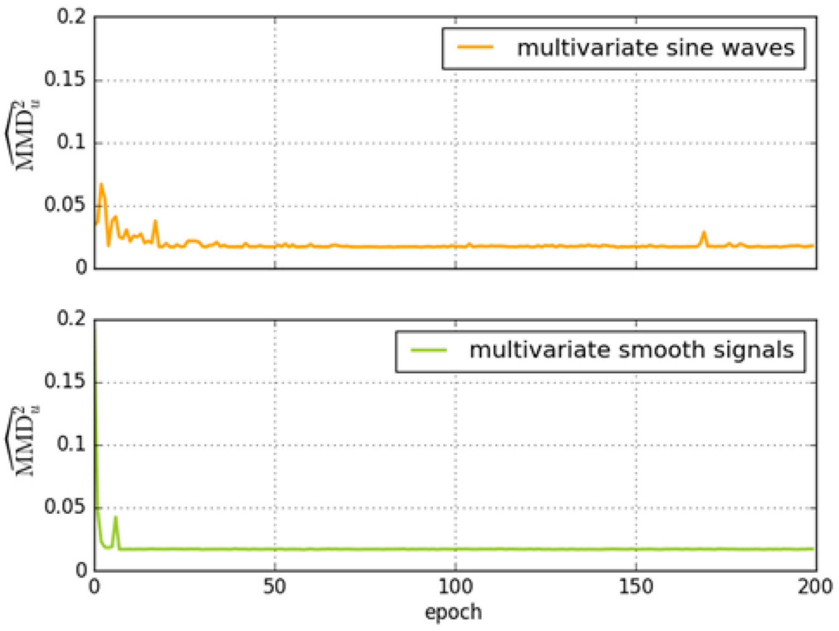

Fig. 9. The traces of $\widehat{\mathrm{MMD}}_u^2$ in the training of MTS-GAN based on the two toy datasets.

$\mathcal{X}^{gen}$ defined above, the distance between their distributions are estimated by the unbiased estimate of the squared MMD (MMD^2) given by [44]:

$$\widehat{\mathrm{MMD}}_u^2 = \frac{\sum_{i=1}^{M_1}\sum_{j\neq i}^{M_1} K(\mathbf{X}_i^{com}, \mathbf{X}_j^{com})}{M_1(M_1-1)} + \frac{\sum_{i=1}^{M_3}\sum_{j\neq i}^{M_3} K(\mathbf{X}_i^{gen}, \mathbf{X}_j^{gen})}{M_3(M_3-1)} - \frac{2\sum_{i=1}^{M_1}\sum_{j=1}^{M_3} K(\mathbf{X}_i^{com}, \mathbf{X}_j^{gen})}{M_1 M_3} \tag{4}$$

where $K : X \times Y \to \mathbb{R}$ is a kernel. Please refer to [44] for detail. The traces of $\widehat{\mathrm{MMD}}_u^2$ in the training of the MTS-GAN based on the two toy datasets are shown in Fig. 9, from which we can find that after several epochs, the $\widehat{\mathrm{MMD}}_u^2$ based on the two datasets basically converge to low and acceptable values, which are about 0.00175 and 0.00170 respectively. This means that the distribution of the synthetic data is very close to that of the original data.

3.3.3. Interpolation based method

Interpolation is often used in the research on the generative models to demonstrate that a generative model does not simply memorize the training data (or in other words, the model does not overfit to the training data) [55]. According to [44], overfitting will make a generator map most points in the latent space to training samples, which implies that the latent space is hierarchically collapsed [38]. Therefore, for an overfitted model, if we sample two points in the latent space and linearly interpolate between them, we will find sharp transitions among the corresponding synthetic samples. The basic steps of the interpolation based evaluation method are:

(i) select two training samples $\mathbf{X}_m^{com}$ and $\mathbf{X}_n^{com}$ that look different;

(ii) search for their "closest" latent encodings $\mathbf{Z}_m^{com}$ and $\mathbf{Z}_n^{com}$ with a trained GAN model by solving the following optimization problem using back-propagation

$$\mathbf{Z}_*^{com} = \underset{\mathbf{z}}{\operatorname{argmin}}\, \mathcal{L}_{com}(\mathbf{z}|\mathbf{X}_*^{com}) \tag{5}$$

where * denotes n or m, and we use the root mean square error (RMSE) as the loss function which is defined by

$$\mathcal{L}_{com}(\mathbf{z}|\mathbf{X}_*^{com}) = \sqrt{\frac{\left\|\mathbf{X}_*^{com} - G(\mathbf{z})\right\|_F^2}{NT}} \tag{6}$$

in which $\|\cdot\|_F$ is the Frobenius norm, N and T are the number of variables of $\mathbf{X}_*^{com}$ and the data length of $\mathbf{X}_*^{com}$ in time respectively, as defined in Section 3.1, $G(\mathbf{z})$ is the output of the trained generator of MTS-GAN, taking $\mathbf{z}$ as the input. In the following paragraphs, we use $\widehat{\mathbf{Z}_*^{com}}$ to represent the estimated value of $\mathbf{Z}_*^{com}$ obtained by solving (5) using back-propagation.

(iii) linearly interpolate between $\widehat{\mathbf{Z}_m^{com}}$ and $\widehat{\mathbf{Z}_n^{com}}$ to obtain $\{\widehat{\mathbf{Z}_i^{com}}\}_{i=1}^{p}$ according to

$$\widehat{\mathbf{Z}_i^{com}} = \widehat{\mathbf{Z}_m^{com}} + \frac{i-1}{p-1}(\widehat{\mathbf{Z}_n^{com}} - \widehat{\mathbf{Z}_m^{com}}) \tag{7}$$

and check the variations among the corresponding synthetic samples $\{G(\widehat{\mathbf{Z}_i^{com}})\}_{i=1}^{p}$.

Figs. 10 and 11 show the interpolation results based on two pairs of original samples in the two toy datasets, in each of which the first and last columns in green show a pair of original training samples $\mathbf{X}_m^{com}$ and $\mathbf{X}_n^{com}$, and the orange lines in the intermediate six columns are the 3-dimensional synthetic samples $\{G(\widehat{\mathbf{Z}_i^{com}})\}_{i=1}^{6}$, where $G(\widehat{\mathbf{Z}_1^{com}})$ and $G(\widehat{\mathbf{Z}_6^{com}})$ are the reconstructed samples of $\mathbf{X}_m^{com}$ and $\mathbf{X}_n^{com}$ respectively, and $\{G(\widehat{\mathbf{Z}_i^{com}})\}_{i=2}^{5}$ are the synthetic samples corresponding to the interpolated latent points $\{\widehat{\mathbf{Z}_i^{com}}\}_{i=2}^{5}$.

It can be seen that in both figures, the interpolation between the pair of the training samples can produce smooth variations in the sample space, which demonstrates that MTS-GAN has not simply memorized the training data but learned a continuous and meaningful latent space.

From the evaluation results shown above, we can conclude that MTS-GAN performs well in modeling MTS distribution, thus it is able to be applied to imputation.

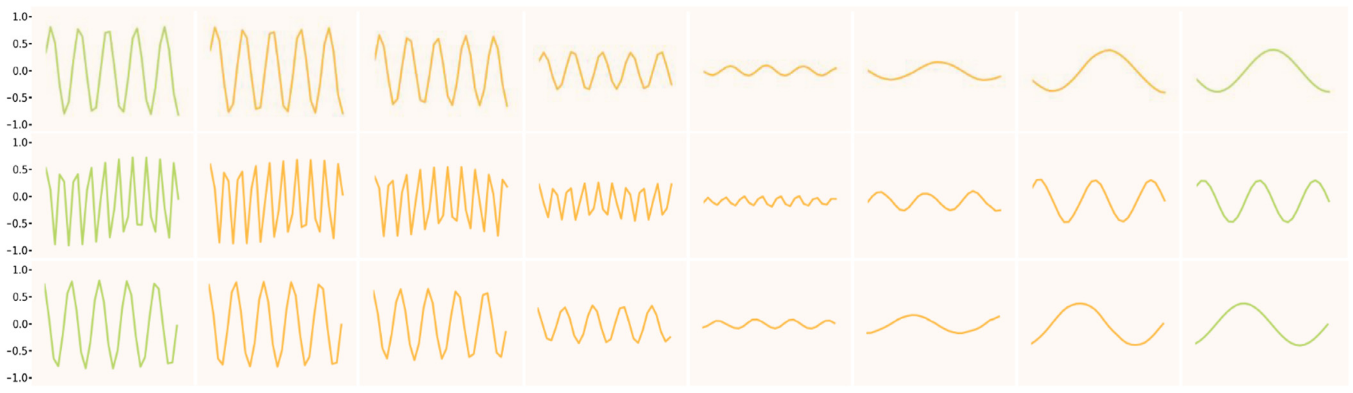

Fig. 10. Interpolation results based on a pair of original samples in toy dataset1.

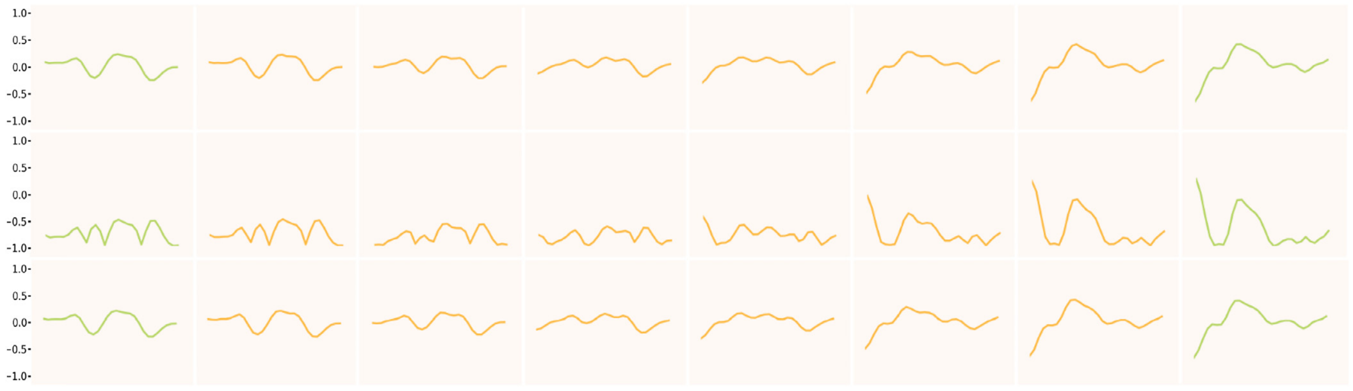

Fig. 11. Interpolation results based on a pair of original samples in toy dataset2.

4. An MTS-GAN based imputation method for MTS

4.1. Methodology

In this section, MTS-GAN is applied to MTS imputation. Considering MTS imputation as a constrained MTS generation task, we propose an MTS-GAN based imputation method. A framework similar to the GAN based image inpainting framework proposed in [39] is used in our method, but there are two differences: (i) The model used in [39] is DC-GAN, but here the proposed MTS-GAN model is adopted; (ii) Reference [39] proposes the weighted sum of two losses as the loss function for semantic image inpainting, in which one loss is based on the generator, and another is based on both the generator and the discriminator. In this work, we discard the discriminator after training and only reserve the generator based loss as the loss function. This avoids introducing additional hyper-parameters (i.e. weights to balance two losses) to the back-propagation.

As described in Section 3.1, given an MTS dataset $\mathcal{X} = \{\mathbf{X}_m\}_{m=1}^{M}$, we first divide all samples into the incomplete subset $\mathcal{X}^{inc} = \{\mathbf{X}_m^{inc}\}_{m=1}^{M_2}$ and the complete subset $\mathcal{X}^{com} = \{\mathbf{X}_m^{com}\}_{m=1}^{M_1}$, and then use $\mathcal{X}^{com}$ to train the MTS-GAN. After that, we apply the trained GAN model to imputation. For the imputation of $\mathcal{X}^{inc}$, we still employ back-propagation to find the "closest" latent encoding of $\mathcal{X}^{inc}$, and then use the samples generated by the generator to impute the missing values. The process of finding the "closest" latent encoding of $\mathcal{X}^{inc}$ in imputation is similar to the second step of the interpolation based evaluation method shown in Section 3.3.3, but it is worth noticing that in imputation, the given samples are incomplete, and we cannot calculate the loss designed by (6) because the loss of truly missing values is unavailable. So in imputation, the loss function should be modified to only involve the existing values as follows:

$$\mathcal{L}_{inc}(\mathbf{z}|\mathcal{X}^{inc}) = \sqrt{\frac{1}{M_2}\sum_{m=1}^{M_2}\frac{\left\|(\mathbf{X}_m^{inc} - G(\mathbf{z}_m))\odot\delta_m^{inc}\right\|_F^2}{sum(\delta_m^{inc})}} \tag{8}$$

and the "closest" latent encoding is formulated as:

$$\mathcal{Z}^{inc} = \underset{\mathbf{z}}{\operatorname{argmin}}\, \mathcal{L}_{inc}(\mathbf{z}|\mathcal{X}^{inc}) \tag{9}$$

where $\mathbf{z} = \{\mathbf{z}_m\}_{m=1}^{M_2}$, δ_m^{inc} is the matrix describing whether each element of $\mathbf{X}_m^{inc}$ is missing or not (see notations in Section 3.1), $\odot$ denotes the element-wise multiplication, $sum(\cdot)$ represents the sum of all the elements in the matrix. After solving (9) with back-propagation, we are able to obtain $\widehat{\mathcal{Z}^{inc}} = \{\widehat{\mathbf{Z}_m^{inc}}\}_{m=1}^{M_2}$. Let $\mathbf{I}$ denote the identity matrix, and then for $m = 1$ to M_2, the reconstructed sample $\mathbf{X}_m^{rec}$ can be calculated as follows:

$$\mathbf{X}_m^{rec} = \mathbf{X}_m^{inc} \odot \delta_m^{inc} + G(\widehat{\mathbf{Z}_m^{inc}}) \odot (\mathbf{I} - \delta_m^{inc}) \tag{10}$$

or in other words,

$$\mathbf{X}_m^{rec}(n,t) = \begin{cases} \mathbf{X}_m^{inc}(n,t), & \text{if } \mathbf{X}_m^{inc}(n,t) \text{ is not missing} \\ G(\widehat{\mathbf{Z}_m^{inc}})(n,t), & \text{if } \mathbf{X}_m^{inc}(n,t) \text{ is missing} \end{cases} \tag{11}$$

The entire algorithm of the MTS imputation method based on MTS-GAN is shown in Algorithm. Note that we train MTS-GAN with d-dimensional latent random vectors $z \sim U([0, 1]^d)$ following uniform distribution as the input, so the trained generator establishes a map from $[0, 1]^d$ to the synthetic data. Therefore, to avoid $\mathbf{z}$ out of range, we need to clip $\mathbf{z}$ to $[0, 1]^d$ after each iteration in back-propagation.

Remark: (6) and (8) are the RMSE losses for finding the closest encodings with back-propagation, but they are used in two different scenarios. (6) is adopted in evaluating the performance of GAN

Algorithm The MTS imputation method based on MTS-GAN.

```
Require: Dataset $\mathcal{X}$, learning rate $\alpha$, and number of iterations $k$ in back-propagation
1: Divide $\mathcal{X}$ into $\mathcal{X}^{com}$ and $\mathcal{X}^{inc}=\{\mathbf{X}_m^{inc}\}_{m=1}^{M_2}$.
2: Take $d$-dimensional latent random vectors $z\sim U([0,1]^d)$ following uniform distribution as the input, and train $G$ and $D$ of the MTS-GAN with $\mathcal{X}^{com}$ by solving the optimization problem defined by (1) and (2).
3: Search for $\mathcal{Z}^{inc}$ by solving (9) with back-propagation:
4: for $k$ iterations do
5:     Calculate the loss $\mathcal{L}_{inc}(\mathbf{z}|\mathcal{X}^{inc})$:
           $\mathcal{L}_{inc}(\mathbf{z}|\mathcal{X}^{inc}) = \sqrt{\frac{1}{M_2}\sum_{m=1}^{M_2}\frac{\left\|(\mathbf{X}_m^{inc}-G(\mathbf{z}_m))\odot\delta_m^{inc}\right\|_F^2}{sum(\delta_m^{inc})}}$
6:     Update $\mathbf{z}$ by descending its gradient:
           $\mathbf{z}\leftarrow\mathbf{z}-\alpha\,\nabla_{\mathbf{z}}\mathcal{L}_{inc}(\mathbf{z}|\mathcal{X}^{inc})$
7:     Clip $\mathbf{z}$ to $[\mathbf{0},\mathbf{1}]$:
           $\mathbf{z}\leftarrow clip(\mathbf{z},\mathbf{0},\mathbf{1})$
8: end for
9: Obtain $\widehat{\mathcal{Z}^{inc}}=\{\widehat{\mathbf{Z}_m^{inc}}\}_{m=1}^{M_2}$ after back-propagation and impute $\{\mathbf{X}_m^{inc}\}_{m=1}^{M_2}$:
10: for $m=1$ to $M_2$ do
11:     The reconstructed sample $\mathbf{X}_m^{rec}$ is calculated by:
           $\mathbf{X}_m^{rec}=\mathbf{X}_m^{inc}\odot\delta_m^{inc}+G(\widehat{\mathbf{Z}_m^{inc}})\odot(\mathbf{I}-\delta_m^{inc})$
12: end for
```

in modeling MTS distribution based on the complete dataset, so (6) is averaged on all elements in the MTS matrix. However, (8) is utilized in the imputation stage based on the incomplete dataset, so it is averaged on only existing values owing to we cannot get the loss of truly missing values.

4.2. Experiments on MTS imputation

In this subsection, we compare the performance of MTS-GAN with the other five existing approaches on the imputation of MTS, including R-GAN based method, DC-GAN based method, DAE based method [14], as well as two interpolation techniques based on spline interpolation and cubic interpolation. Besides, we also investigate the imputation accuracy under different missing rates to illustrate the robustness of each method. Three simulated datasets and a real-world dataset are used in the experiments.

4.2.1. Description of the datasets

1) Toy dataset3: As toy dataset1 in Section 3.3, toy dataset3 are the multivariate sine waves generated according to

$$\left[A\sin(2\pi ft+\phi)\ A\sin(4\pi ft+2\phi)\ A\cos(2\pi ft+\phi)\right]^T$$

where A, f, ϕ for different samples are randomly generated and belong to [0.1,0.9], [1.0,5.0], $[-\pi,\pi]$, respectively. The dataset contains 8400 training samples and 2800 testing samples.

2) Toy dataset4: In this dataset, each dimension is generated by the superposition of sine waves of different frequencies as follows:

$$\begin{bmatrix}A\sin(2\pi ft+\phi)+A\cos(2\pi ft+\phi)\\ A\sin(3\pi ft+\phi)+A\cos(2\pi ft+\phi)\\ A\sin(2\pi ft+\phi)+A\cos(3\pi ft+\phi)\end{bmatrix}$$

where A, f, ϕ for different samples are randomly generated and belong to [0.1,0.9], [1.0,5.0], $[-\pi,\pi]$, respectively. The dataset also contains 8400 training samples and 2800 testing samples.

3) Tennessee Eastman Process (TEP) based dataset: TEP model [56] is used to generate the dataset, which is built from an actual chemical process by Downs and Vogel in 1993, and is a famous industrial process simulation platform in the fields of process control, process monitoring and fault diagnosis. In this part, the TEP based dataset is generated by a Simulink code running at the base case which can be downloaded from http://depts.washington.edu/control/LARRY/TE/download.html. In the experiments, ten continuous variables (XMEAS(1), (7), (9), (10), (11), (13), (16), (18), (20), (21)) are considered for four scenarios, i.e. IDV(0), IDV(1), IDV(2), IDV(7). The dataset contains 3200 training samples and 800 testing samples in total, and each sample is a 10-dimensional time series of length 50.

4) Point machine dataset: A point machine is a device for operating railway turnouts to determine the heading direction of a train when facing railway crossings. Since the operating currents are important features for fault detection of point machine operations, historical dataset collected from a real-world high-speed railway in China, which is composed of three-phase operating currents of point machines, is used as the dataset in this subsection. In this dataset, there are a total of 9000 training samples and 1000 testing samples, and each of them is a 3-dimensional time series of length 140.

4.2.2. Experiment settings

For each dataset, the training samples are used to train GAN and DAE (normalized to $[-1,1]$ before training), while the testing samples are used to produce incomplete dataset for checking the performance of imputation in the evaluation stage. For each dataset mentioned in Section 4.2.1, we denote the training set and the testing set as $\mathcal{X}^{train}=\{\mathbf{X}_m^{train}\}_{m=1}^{M_1}$ and $\mathcal{X}^{test}=\{\mathbf{X}_m^{test}\}_{m=1}^{M_2}$ respectively, and produce the incomplete dataset $\mathcal{X}^{test-inc}=\{\mathbf{X}_m^{test-inc}\}_{m=1}^{M_2}$ by randomly setting some elements of $\mathcal{X}^{test}$ to be missing in the following ways:

(1) Random missing case (case 1): For each sample of $\mathcal{X}^{test}$, i.e. $\mathbf{X}_m^{test}$, $m=1,2,\ldots,M_2$, we randomly set p% of its elements to be missing, where p is a random integer ranging from p_1 to p_2. So in the incomplete dataset $\mathcal{X}^{test-inc}$, each sample has a missing rate ranging from p_1% to p_2%.

(2) Continuous missing case (case 2): For each dimension j of $\mathbf{X}_m^{test}$, $m=1,2,\ldots,M_2$, we randomly select a length i ranging from $p_1\%\times T$ to $p_2\%\times T$ and a starting instant t_0 ranging from 1 to $T-i$ respectively, and then set $\mathbf{X}_m^{test}(j,t_0),\mathbf{X}_m^{test}(j,t_0+1),\ldots,\mathbf{X}_m^{test}(j,t_0+i)$ to be missing.

In the evaluation stage, we use the error defined by (12) as the measure to evaluate the performance of imputation:

$$error=\sqrt{\frac{1}{M_2}\sum_{m=1}^{M_2}\frac{\left\|(\mathbf{X}_m^{test}-\mathbf{X}_m^{test-rec})\odot(\mathbf{I}-\delta_m^{test-inc})\right\|_F^2}{sum(\mathbf{I}-\delta_m^{test-inc})}} \tag{12}$$

where $\mathbf{X}_m^{test-rec}$, $m=1,2,\ldots,M_2$ is the reconstructed complete sample corresponding to $\mathbf{X}_m^{test-inc}$, $m=1,2,\ldots,M_2$ obtained by imputation.

The discriminator of MTS-GAN adopted in the experiment is composed of two strided convolutional layers with 1-D kernels and three fully-connected layers. For convenience, we describe the discriminator of MTS-GAN as $C_1(64,3)-C_2(128,3)-F_1(1024)-F_2(256)-F_3(1)$, in which $C_x(y,z)$ denotes that y kernels are used in the x-th convolutional layer with a size of z, and similarly, $F_x(y)$ represents that the number of nodes in the

Table 1
The reconstruction error of each approach on the four datasets, in which the missing rate interval $[p_1\%, p_2\%]$ is set to [10%, 90%]. **Bold** numbers are the best results.

Datasets	Toy dataset3		Toy dataset4		TEP based dataset		Point machine dataset	
Methods	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2
MTS-GAN based method	**0.0241**	**0.0244**	**0.0145**	**0.0143**	**0.0212**	**0.0225**	**0.0390**	**0.0303**
DC-GAN based method	0.1569	0.1788	0.1458	0.1633	0.0222	0.0264	0.0400	0.0307
R-GAN based method	0.0388	0.0414	0.0240	0.0289	0.0630	0.0650	0.1047	0.0886
DAE based method [13]	0.0675	0.1037	0.0465	0.0705	0.0322	0.0920	0.0488	0.0417
Cubic interpolation	0.4469	0.6628	0.3631	0.5708	0.0788	0.3407	0.1324	0.2307
Spline interpolation	0.4754	0.8181	0.4008	0.7344	0.1151	0.4569	0.1809	0.5089

Table 2
The reconstruction error of each approach on the four datasets, in which the missing rate interval $[p_1\%, p_2\%]$ is set to [30%, 50%]. **Bold** numbers are the best results.

Datasets	Toy dataset3		Toy dataset4		TEP based dataset		Point machine dataset	
Methods	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2
MTS-GAN based method	**0.0232**	**0.0225**	**0.0140**	**0.0138**	**0.0207**	**0.0223**	0.0330	**0.0253**
DC-GAN based method	0.1489	0.1601	0.1393	0.1439	0.0214	0.0255	**0.0300**	0.0254
R-GAN based method	0.0322	0.0354	0.0211	0.0257	0.0611	0.0615	0.1018	0.0841
DAE based method [13]	0.0396	0.0606	0.0262	0.0358	0.0235	0.0625	0.0410	0.0366
Cubic interpolation	0.3618	0.6125	0.2561	0.5200	0.0370	0.2251	0.1061	0.1593
Spline interpolation	0.3629	0.7383	0.2484	0.6426	0.0730	0.2991	0.1629	0.3692

x-th fully connected layer is y. In the same way, the generator can be expressed as $F_1(256) - F_2(1024) - F_3(N \times 128 \times \lceil \frac{T}{4} \rceil) - D_1(64, 3) - D_2(1, 3)$, where $D_x(y, z)$ denotes the parameters of the fractional-strided convolutional layers, $\lceil \cdot \rceil$ is the ceiling function, N and T are the number of variables of the input MTS and the data length of each MTS in time respectively. In addition, the input of the generator are $N \times 10$ dimensional latent random vectors following a uniform distribution. Note that in both strided convolutional layers and fractional-strided convolutional layers, each kernel convolutes its input with stride 2.

4.2.3. Experimental results

In this paper, all of the networks in the experiment are implemented based on Tensorflow [57] framework. The experiment is repeated ten times on each dataset and the reported results in all tables are the averaged reconstruction error.

It is worth mentioning that in the experiment, the optimization of back-propagation procedure is easy to get stuck in sub-optimal critical points and $\widehat{\mathcal{Z}^{inc}}$ is very sensitive to the initial values, both of which make it difficult to find the "closest" latent encodings of incomplete samples precisely. To reduce the influence of two points mentioned above, in each experiment, we perform back-propagation procedure for multiple times with different initial values, and finally impute each incomplete sample with its best encoding which has the lowest loss (8). Furthermore, we clip the results of interpolation (both cubic and spline interpolation) to $[-1, 1]$ because in some time series, the last several values are missing successively, which may lead to a very large imputation error.

Table 1 shows the imputation results of the five existing approaches and the proposed MTS-GAN based method tested on the four datasets under the two different missing cases (i.e. random missing and continuous missing cases) when the missing rate interval $[p_1\%, p_2\%]$ is set to [10%, 90%]. It can be seen from Table 1 that the MTS-GAN based imputation method achieves the best reconstruction accuracy compared with other approaches under both the random missing and continuous missing cases.

Figs. 12 and 13 give two examples of samples in $\mathcal{X}^{test}$, $\mathcal{X}^{test\text{–}inc}$ based on the TEP based dataset and point machine dataset respectively, as well as their corresponding reconstructed samples obtained by the proposed approach. We can clearly observe from the

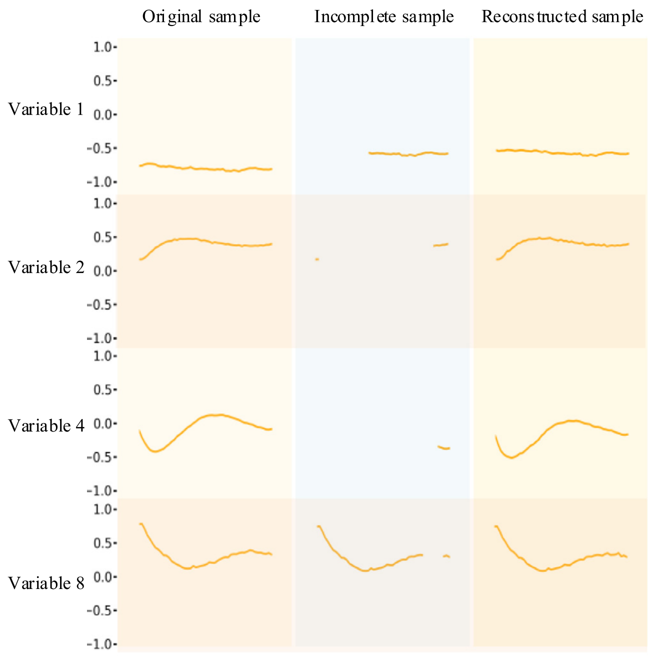

Fig. 12. An example of samples in $\mathcal{X}^{test}$, $\mathcal{X}^{test\text{–}inc}$ based on the TEP based dataset, and its corresponding reconstructed sample obtained by the proposed approach.

two figures that the proposed MTS-GAN successfully estimates the missing values.

Besides a high imputation accuracy, a good imputation method should be robust to the missing rate. To investigate how the missing rate affects the imputation accuracy of each method, we carry out three sets of experiments under different missing rates, in which the missing rate intervals $[p_1\%, p_2\%]$ are set to [30%, 50%], [50%, 70%], [70%, 90%] respectively. Tables 2–4 show the imputation results of the three sets of experiments respectively, from which we can observe that the higher the missing rate, the lower the imputation accuracy, which is a very intuitive and easy to understand result. In addition, as shown in the three tables, the proposed MTS-GAN imputation method achieves the best performance in almost

Table 3
The reconstruction error of each approach on the four datasets, in which the missing rate interval [p_1%, p_2%] is set to [50%, 70%]. **Bold** numbers are the best results.

Datasets	Toy dataset3		Toy dataset4		TEP based dataset		Point machine dataset	
Methods	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2
MTS-GAN based method	**0.0237**	**0.0241**	**0.0144**	**0.0145**	0.0214	**0.0226**	**0.0375**	**0.0280**
DC-GAN based method	0.1530	0.2091	0.1433	0.1883	0.0222	0.0272	0.0397	0.0310
R-GAN based method	0.0355	0.0505	0.0226	0.0318	0.0628	0.0688	0.1051	0.0874
DAE based method [13]	0.0380	0.1069	0.0267	0.0841	**0.0197**	0.0507	0.0422	0.0354
Cubic interpolation	0.4638	0.6556	0.3690	0.5741	0.0684	0.2886	0.1324	0.1987
Spline interpolation	0.4977	0.8325	0.4104	0.7488	0.1127	0.3903	0.1921	0.4534

Table 4
The reconstruction error of each approach on the four datasets, in which the missing rate interval [p_1%, p_2%] is set to [70%, 90%]. **Bold** numbers are the best results.

Datasets	Toy dataset3		Toy dataset4		TEP based dataset		Point machine dataset	
Methods	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2
MTS-GAN based method	**0.0256**	**0.0327**	**0.0154**	**0.0182**	**0.0222**	**0.0234**	**0.0452**	**0.0423**
DC-GAN based method	0.1733	0.2994	0.1626	0.2717	0.0242	0.0285	0.0569	0.0509
R-GAN based method	0.0509	0.1052	0.0294	0.0491	0.0678	0.0779	0.1108	0.1011
DAE based method [13]	0.1085	0.3028	0.0827	0.2620	0.0430	0.0498	0.0554	0.0580
Cubic interpolation	0.6086	0.7300	0.5252	0.6499	0.1536	0.4517	0.1659	0.3099
Spline interpolation	0.6840	0.9178	0.6037	0.8552	0.1931	0.5853	0.2252	0.6342

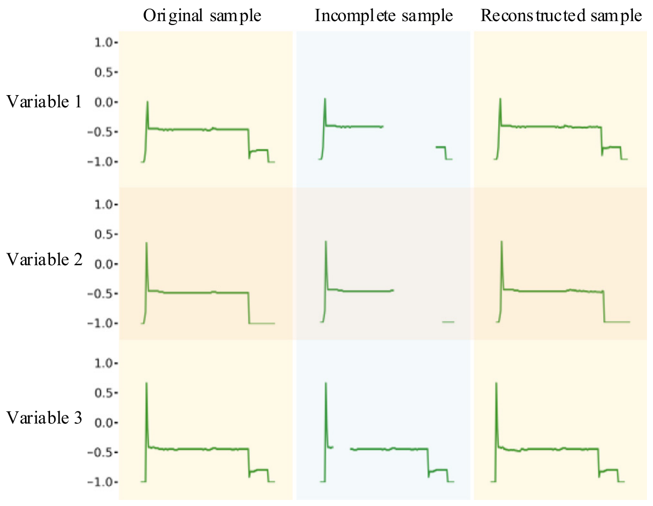

Fig. 13. An example of samples in $\mathcal{X}^{test}$, $\mathcal{X}^{test\text{-}inc}$ based on the point machine dataset, and its corresponding reconstructed sample obtained by the proposed approach.

all cases. Although in two specific cases (i.e., case 1 of the point machine dataset in Tables 2 as well as case 1 of the TEP based dataset in Tables 3), the proposed MTS-GAN fails to give the best performance, it indeed obtains the second smallest reconstruction error which is very close to the best one. More importantly, we are able to find that compared with the other five approaches, the imputation accuracy of our method is least affected by the missing rate, which indicates that the proposed imputation method performs more robustly as the missing rate increases.

The main reason for MTS-GAN to perform well in modeling MTS distribution and further to achieve the best performance in MTS imputation is that MTS-GAN is mainly proposed by introducing the multi-channel idea of MC-CNN [47] into DC-GAN [38], and it combines the strong abilities of DC-GAN in modeling distribution with the superior abilities of multi-channel convolution in processing MTS. On one hand, the DC-GAN architecture enables MTS-GAN to capture data distribution in a better way than the other imputation methods. On the other hand, unlike DC-GAN which adopts 2-D convolutions, MTS-GAN utilizes multi-channel convolution specially designed for MTS to perform 1-D kernel based convolutions in each single channel, which makes MTS-GAN have a better performance than DC-GAN in processing MTS.

5. Conclusion

In this paper, to improve MTS imputation performance, we propose a new variant of GANs, i.e. MTS-GAN, for MTS distribution modeling, and further apply it to MTS imputation by regrading imputation as a constrained MTS generation task. Simulation results based on two toy datasets show that MTS-GAN performs well in capturing the distribution of MTS. Finally, experimental results based on three simulated datasets and a real-world dataset composed of historical field data collected from a high-speed railway show that compared with several approaches, the proposed MTS-GAN based imputation method not only achieves a higher imputation accuracy under different missing rates, but also performs more robustly as the missing rate increases. Future work includes the industrial fields application of the proposed MTS imputation method. MTS Imputation will be beneficial to the data mining of the historical data such as fault detection and fault diagnosis — because in the data pre-processing stage, incomplete samples are often discarded, which lead to the loss of all of their information, while imputation reserves part of the information of original incomplete samples, which can contribute to the classification and clustering.

Conflicts of interest

The authors declare that they have no conflicts of interest to this work.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant 2017YFB1200700, the National Natural Science Foundation of China under Grant 61490701, 61803163, the special fund of Suzhou-Tsinghua Innovation Leading Action under Grant 2016SZ0202 and the Research and Development Project of Beijing National Railway Research & Design Institute of Signal & Communication Ltd.

References

[1] B.R. Bakshi, Multiscale PCA with application to multivariate statistical process monitoring, AIChE J. 44 (7) (1998) 1596–1610.
[2] F. Serdio, et al., Fault detection in multi-sensor networks based on multivariate time-series models and orthogonal transformations, Inf. Fusion 20 (2014) 272–291.
[3] P.T. Theodossiou, Predicting shifts in the mean of a multivariate time series process: an application in predicting business failures, J. Am. Stat. Assoc. 88 (422) (1993) 441–449.
[4] E. Kova-Andri, J. Brana, V. Gvozdi, Impact of meteorological factors on ozone concentrations modelled by time series analysis and multivariate statistical methods, Ecol. Inform. 4 (2) (2009) 117–122.
[5] M. Palu, Detecting nonlinearity in multivariate time series, Phys. Lett. A 213 (3–4) (1996) 138–147.
[6] U. Gather, M. Imhoff, R. Fried, Graphical models for multivariate time series from intensive care monitoring, Stat. Med. 21 (18) (2002) 2685–2701.
[7] J. Barnard, X.L. Meng, Applications of multiple imputation in medical studies: from AIDS to NHANES, Stat. Methods Med. Res. 8 (1) (1999) 17–36.
[8] A. Farhangfar, L.A. Kurgan, W. Pedrycz, A novel framework for imputation of missing values in databases, IEEE Trans. Syst., Man, Cybern.-Part A: Syst. Hum. 37 (5) (2007) 692–709.
[9] G.L. Schlomer, S. Bauman, N.A. Card, Best practices for missing data management in counseling psychology, J. Counsel. Psychol. 57 (1) (2010) 1.
[10] T.D. Pigott, A review of methods for missing data, Educ. Res. Eval. 7 (4) (2001) 353–383.
[11] O. Troyanskaya, et al., Missing value estimation methods for DNA microarrays, Bioinformatics 17 (6) (2001) 520–525.
[12] E.D. Schneiderman, C.J. Kowalski, S.M. Willis, Regression imputation of missing values in longitudinal data sets, Int. J. Bio-med. Comput. 32 (2) (1993) 121–133.
[13] H.H. Hsu, A.C. Yang, M.D. Lu, KNN-DTW based missing value imputation for microarray time series data, JCP 6 (3) (2011) 418–425.
[14] L. Gondara, K. Wang, Multiple imputation using deep denoising autoencoders, 2017. arXiv preprint arXiv:1705.02737.
[15] W.G. Madow, Incomplete Data in Sample Surveys, Academic Press, 1983.
[16] P.L. Roth, Missing data: a conceptual review for applied psychologists, Pers. Psychol. 47 (3) (1994) 537–560.
[17] R.J. Little, D.B. Rubin, Statistical Analysis with Missing Data, John Wiley & Sons, 2014.
[18] A.S. Dhevi, Imputing missing values using inverse distance weighted interpolation for time series data, in: 2014 Sixth International Conference on Advanced Computing (ICoAC), IEEE, 2014.
[19] N.F.A. Radi, R. Zakaria, M.A.z. Azman, Estimation of missing rainfall data using spatial interpolation and imputation methods, in: AIP Conference Proceedings, AIP, 2015.
[20] T.E. Raghunathan, et al., A multivariate technique for multiply imputing missing values using a sequence of regression models, Surv. Methodol. 27 (1) (2001) 85–96.
[21] N. Ankaiah, V. Ravi, A novel soft computing hybrid for data imputation, in: Proceedings of the 7th International Conference on Data Mining (DMIN), 2011.
[22] F. Honghai, et al., A SVM regression based approach to filling in missing values, in: International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, Springer, 2005.
[23] P.D. Allison, Handling missing data by maximum likelihood, in: SAS Global Forum, Statistical Horizons Haverford, PA, USA, 2012.
[24] P. Meesad, K. Hengpraprohm, Combination of knn-based feature selection and knnbased missing-value imputation of microarray data, in: 3rd International Conference on Innovative Computing Information and Control (ICICIC'08), IEEE, 2008.
[25] W. Ling, F. Dong-Mei, Estimation of missing values using a weighted k-nearest neighbors algorithm, in: Environmental Science and Information Application Technology, IEEE, 2009. ESIAT 2009. International Conference on. 2009
[26] K. Wellenzohn, M.H. Böhlen, A. Dignös, J. Gamper, H. Mitterer, Continuous imputation of missing values in streams of pattern-determining time series, in: EDBT, 2017, pp. 330–341.
[27] X. Gao, Y. Wang, Application of EM algorithm in statistics natural language processing, Res. J. Appl. Sci. Eng. Tech. (10) (2013) 2969–2973.
[28] S. Sridevi, et al., Imputation for the analysis of missing values and prediction of time series data, in: 2011 International Conference on Recent Trends in Information Technology (ICRTIT), IEEE, 2011.
[29] L. Beretta, A. Santaniello, Nearest neighbor imputation algorithms: a critical evaluation, BMC Med. Inform. Decis. Making 16 (3) (2016) 74.
[30] Y. Bengio, F. Gingras, Recurrent neural networks for missing or asynchronous data, in: Advances in Neural Information Processing Systems, 1996.
[31] Z. Che, S. Purushotham, K. Cho, D. Sontag, Y. Liu, Recurrent neural networks for multivariate time series with missing values, Scientific reports 8 (1) (2018) 6085.
[32] N. Jaques, et al., Multimodal Autoencoder: A Deep Learning Approach to Filling In Missing Sensor Data and Enabling Better Mood Prediction.
[33] I. Goodfellow, et al., Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014.
[34] I. Goodfellow, NIPS 2016 tutorial: generative adversarial networks, 2016. arXiv preprint arXiv:1701.00160.
[35] A. Gupta, et al., Social GAN: Socially acceptable trajectories with generative adversarial networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[36] Z. Yi, et al., DualGAN: Unsupervised dual learning for image-to-image translation, in: ICCV, 2017.
[37] Y. Lu, Y.W. Tai, C.K. Tang, Conditional cyclegan for attribute guided face image generation, 2017. arXiv preprint arXiv:1705.09966.
[38] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. arXiv preprint arXiv:1511.06434.
[39] R. Yeh, C. Chen, T.Y. Lim, M. Hasegawa-Johnson, M.N. Do, Semantic image inpainting with perceptual and contextual losses, 2016. arXiv preprint arXiv:1607.07539.
[40] Y. Li, S. Liu, J. Yang, M. H. Yang, Generative face completion, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3911–3919.
[41] M. Chidambaram, Y. Qi, Style transfer generative adversarial networks: learning to play chess differently, 2017. arXiv preprint arXiv:1702.06762.
[42] J.Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2223–2232.
[43] J.P. Assendorp, in: Deep learning for anomaly detection in multivariate time series data, 2017. Hochschule fr Angewandte Wissenschaften Hamburg
[44] C. Esteban, S.L. Hyland, G. Rtsch, Real-valued (medical) time series generation with recurrent conditional gans, 2017. arXiv preprint arXiv:1706.02633.
[45] N.R. Ke, A. Goyal, O. Bilaniuk, J. Binas, L. Charlin, C. Pal, Y. Bengio, Sparse attentive backtracking: long-range credit assignment in recurrent networks, 2017. arXiv preprint arXiv:1711.02326.
[46] Y. LeCun, et al., Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[47] Y. Zheng, et al., Exploiting multi-channels deep convolutional neural networks for multivariate time series classification, Front. Comput. Sci. 10 (1) (2016) 96–112.
[48] J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: the all convolutional net, 2014. arXiv preprint arXiv:1412.6806.
[49] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proceedings of the ICML, 2013.
[50] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010.
[51] T. Salimans, et al., Improved techniques for training gans, in: Advances in Neural Information Processing Systems, 2016.
[52] A. Gretton, et al., A kernel method for the two-sample-problem, in: Advances in Neural Information Processing Systems, 2007.
[53] S. Xiang, H. Li, On the effects of batch and weight normalization in generative adversarial networks, 2017. arXiv preprint arXiv:1704.03971.
[54] A. Borji, Pros and cons of gan evaluation measures, Computer Vision and Image Understanding 179 (2019) 41–65.
[55] T. White, Sampling generative networks, 2016. arXiv preprint arXiv:1609.04468.
[56] J.J. Downs, E.F. Vogel, A plant-wide industrial process control problem, Comput. Chem. Eng. 17 (3) (1993) 245–255.
[57] M. Abadi, et al., Tensorflow: a system for large-scale machine learning, in: OSDI, 2016.

Zijian Guo received the B.Eng. degree in the school of information science and engineering from Central South University, Changsha, Hunan, China, in 2015. He has been working towards the Ph.D. degree in the department of Automation, Tsinghua University, Beijing, China, since 2015. His current research interests include neural networks and machine learning, as well as their applications in fault detection and fault diagnosis.

Yiming Wan is an Associate Professor in the School of Automation, Huazhong University of Science and Technology, Wuhan, China. He received the Ph.D. degree in Control Science & Engineering from Tsinghua University in 2013. From 2013 to 2016, he was a postdoc researcher at the Delft University of Technology, The Netherlands. During 2016–2018, he was a Research Associate at Massachusetts Institute of Technology, the United States. His research interests include fault diagnosis and fault-tolerant model predictive control with applications in aerospace, energy, and industrial processes.

Hao Ye was born in China, in 1969. He received his Bachelor and Doctoral Degrees in Automation from Tsinghua University, China, in 1992 and 1996 respectively. He has been with the Department of Automation, Tsinghua University since 1996. He is currently a professor and the director of the Institute of Process Control Engineering of the Department of Automation of Tsinghua University. He is mainly interested in fault detection and diagnosis of dynamic systems.