Generative Semi-supervised Learning for Multivariate Time Series Imputation

Xiaoye Miao[1], **Yangyang Wu**[1], **Jun Wang**[3], **Yunjun Gao**[2], **Xudong Mao**[4], **Jianwei Yin**[1,2*]
[1] Center for Data Science, Zhejiang University, Hangzhou, China
[2] College of Computer Science, Zhejiang University, Hangzhou, China
[3] Information Hub, The Hong Kong University of Science and Technology, Hong Kong, China
[4] Department of Artificial Intelligence, Xiamen University, Xiamen, China
{miaoxy, zjuwuyy, junw, gaoyj}@zju.edu.cn, xudong.xdmao@gmail.com, zjuyjw@cs.zju.edu.cn

Abstract

The missing values, widely existed in multivariate time series data, hinder the effective data analysis. Existing time series imputation methods do not make full use of the label information in real-life time series data. In this paper, we propose a novel semi-supervised generative adversarial network model, named SSGAN, for missing value imputation in multivariate time series data. It consists of three players, i.e., a *generator*, a *discriminator*, and a *classifier*. The classifier predicts labels of time series data, and thus it drives the generator to estimate the missing values (or components), conditioned on observed components and data labels at the same time. We introduce a *temporal reminder matrix* to help the discriminator better distinguish the observed components from the imputed ones. Moreover, we theoretically prove that, SSGAN using the temporal reminder matrix and the classifier does learn to estimate missing values converging to the true data distribution when the Nash equilibrium is achieved. Extensive experiments on three public real-world datasets demonstrate that, SSGAN yields a more than 15% gain in performance, compared with the state-of-the-art methods.

Introduction

Multivariate time series analysis (Shi et al. 2020; Song et al. 2018; Wang et al. 2018) is of great practicality in real-life scenarios, such as stock price forecasting (Bai and Ng 2008), health status predicting of patients (Esteban, Hyland, and Rätsch 2017), and weather forecasting (Bright et al. 2015). *Incomplete* time series data are *ubiquitous* in many applications (Miao et al. 2016, 2019), including medical logs, meteorology records, and traffic data, due to various reasons of the collection device failure, instable system environment, or privacy concerns. For example, the public real-world medical time series dataset PhysioNet takes above 80% average missing rate, making it difficult to analyze and mine (Silva et al. 2012). As a result, the missing data problem seriously hinders the effective analysis of time series data.

Data imputation has been extensively explored to solve the missing problem. Traditional methods (Mattei and Frellsen 2019; Yoon, Jordon, and Schaar 2018) for *static* data imputation have limited ability to impute missing values in time series data, since they disregard the relationship between the variables over time. In contrast, the time series imputation approaches (Fortuin et al. 2019; Liu et al. 2019; Luo et al. 2018, 2019; Ma et al. 2019), consider the *temporal* information in multivariate time series data to get better time series imputation accuracy.

The ultimate goal of imputing missing values on multivariate time series data is to benefit the *downstream data analytics*, e.g., classification. In practice, the time series datasets generally more or less take *some labels* in many real-life scenarios. As an example, the PhysioBank archive (Goldberger et al. 2000) contains more than 40 gigabytes of ECG medical time series data with some missing values and a tiny subset of annotated labels. Hospitals often archive even larger amounts of ECG data *without* labels for legal reasons.

Accordingly, it drives to integrate the time series imputation and subsequent analysis. It is also promising and practical to make full use of valuable annotated labels in time series data for missing value imputation in real-life applications. However, almost all existing time series imputation approaches (Fortuin et al. 2019; Liu et al. 2019; Luo et al. 2018, 2019; Ma et al. 2019) do not consider the downstream analysis when imputing the missing values in time series data with a fraction of labels.

Therefore, in this paper, we propose a novel *semi-supervised generative adversarial network* model, termed as SSGAN, to impute missing values in the *general* time series data with a fraction of labels. SSGAN is composed of three players, i.e., a generator, a discriminator, and a classifier. It adopts a *bidirectional* recurrent neural network to learn the temporal information in multivariate time series. We present a semi-supervised classifier in SSGAN to fully use the labels existed in multivariate time series data, so as to take into consideration the subsequent time series analysis task at the same time. The generator estimates the missing values conditioned on observed values/components and data labels in time series. It outputs an imputed time series dataset. The discriminator attempts to identify which components are observed and which ones are imputed by the generator. We arm the discriminator with a *temporal reminder matrix* that encodes a proportion of missing information in the original time series data. Using the temporal reminder matrix and the classifier forces the generator in SSGAN to accurately learn the desirable true data distribution. In a nutshell, the main

[*]Corresponding author: zjuyjw@cs.zju.edu.cn

contributions are summarized as follows.

- We propose a novel semi-supervised generative adversarial network model SSGAN, with three players (i.e., a generator, a discriminator, and a classifier) to predict missing values in the partially labeled time series data.
- We develop a semi-supervised classifier to iteratively classify the *unlabeled* time series data. It drives the generator to estimate missing values using observed components as well as data labels. We introduce a temporal reminder matrix to assist the discriminator to better distinguish the observed components from the imputed ones.
- We prove that, SSGAN using the temporal reminder matrix and the classifier does learn the *true* data distribution.
- Extensive experiments on three public real-world time series datasets demonstrate that, SSGAN substantially outperforms state-of-the-art methods.

Related Work

In this section, we overview existing time series imputation methods, including *statistical* ones and *deep learning* ones.

The statistical time series imputation methods are simple and relatively low-effective. They substitute missing values with the statistics, e.g., zero, mean, and last observed value (Amiri and Jensen 2016), or simple statistical models, including ARIMA (Bartholomew 1971), ARFIMA (Hamzaçebi 2008), and SARIMA (Hamzaçebi 2008), which eliminate the non-stationary parts in the time series data and fit a parameterized stationary model.

Some deep learning models have been employed to impute time series data, including recurrent neural networks (RNN) (Che et al. 2018; Yoon, Zame, and van der Schaar 2017) and deep generative models (Fortuin et al. 2019; Ma et al. 2019). In particular, a gated recurrent unit based time series imputation method (GRUD) (Che et al. 2018) uses a hidden state decay to capture the past features. A cross-dimensional self-attention model (CDSA) (Ma et al. 2019) imputes missing values with location information in multivariate geo-tagged time series data. The time series imputation model (GP-VAE) (Fortuin et al. 2019) predicts missing values in multivariate time series data, via combining variational autoencoder model with Gaussian process.

In contrast, the two-stage GAN imputation method (TSGAN) (Luo et al. 2018) learns the distribution of the observed multivariate time series data to optimize the input vectors of the generator. Then, an end-to-end GAN imputation model E^2GAN (Luo et al. 2019) is further proposed, where the generator takes an autoencoder module to avoid the "noise" vector optimization stage in TSGAN. In addition, a non-autoregressive multi-resolution imputation model (NAOMI) (Liu et al. 2019) imputes missing values in the time series data with adversarial training. However, *none* of the aforementioned imputation methods combines the tasks of time series imputation and subsequent analysis, or utilizes the labels existed in real-life time series data.

Our closest work is BRITS, a *supervised* time series imputation method (Cao et al. 2018). Its model assumes that, *all labels* of times series data are complete, not missing. The training of BRITS completely relies on labeled samples, while the unlabeled samples are discarded during training. It is important to note that, it makes BRITS less-effective for real-life time series data (that generally have a proportion of labels in practice). It is because, for the partially labeled times series data, the number of samples (with labels) used to train BRITS dramatically drops, easily making BRITS overfitting. In contrast, our imputation model SSGAN fully utilizes all (labeled and unlabeled) time series samples during training. Moreover, we iteratively predict labels via a semi-supervised classifier. It further optimizes the imputation, as confirmed in experimental evaluation.

Preliminaries

The input multivariate time series dataset contains a set of samples $\mathbf{S} = (\mathbf{X}_1, \cdots, \mathbf{X}_s)$ with labels $\mathbf{y} = (y_1, \cdots, y_s)$. Each sample in $\mathbf{S}$, i.e., $\mathbf{X}$, is a time series matrix observed at the timestamps $\mathbf{T} = (t_1, \cdots, t_n)$ with a label y. Formally, $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_i, \cdots, \mathbf{x}_n) \in \mathbb{R}^{d\times n}$ with $\mathbf{x}_i$ being $(x_i^1, \cdots, x_i^j, \cdots, x_i^d)^\top$. In particular, x_i^j is the j-th feature value of $\mathbf{X}$ at timestamp t_i, which is probably missing in the original input multivariate time series dataset.

For each sample $\mathbf{X}$, we define a mask matrix $\mathbf{M} = (\mathbf{m}_1, \cdots, \mathbf{m}_i, \cdots, \mathbf{m}_n) \in \mathbb{R}^{d\times n}$ to encode the state of missing values in $\mathbf{X}$, which indicates whether the values in $\mathbf{X}$ exist or not. In particular, $\mathbf{m}_i = (m_i^1, \cdots, m_i^j, \cdots, m_i^d)^\top$, and m_i^j being 0 or 1 means x_i^j is missing or observed.

In time series data, some variables might be missing during consecutive timestamps. Hence, for each sample $\mathbf{X}$, we define a time-lag matrix $\boldsymbol{\delta} = (\boldsymbol{\delta}_1, \cdots, \boldsymbol{\delta}_i, \cdots, \boldsymbol{\delta}_n) \in \mathbb{R}^{d\times n}$ to record the time lag between the current and last timestamp with an observed value, and $\boldsymbol{\delta}_i = (\delta_i^1, \cdots, \delta_i^j, \cdots, \delta_i^d)^\top$. The calculation of the time-lag matrix $\boldsymbol{\delta}$ is defined below.

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

Given an incomplete multivariate time series dataset $\mathbf{S}$, the time series imputation problem studied in this paper aims to find an appropriate value for each missing component in $\mathbf{S}$, so as to (i) make the imputed time series dataset $\hat{\mathbf{X}}$ as close to the real complete dataset $\mathbf{X}$ (if it exists) as possible, and (ii) assist downstream prediction tasks to achieve better performance if adopting the imputed time series dataset than that only using the original incomplete time series dataset.

SSGAN Overview

In this section, we introduce the general framework of the proposed SSGAN model for time series data imputation.

Figure 1 illustrates the general architecture of SSGAN. The input data of SSGAN includes the time series data, (incomplete) data-label pairs, mask matrix, and time-lag matrix. SSGAN consists of three players, including a generator G, a discriminator D, and a classifier C. The generator G estimates the missing components depending on observed components and labels in time series data. It outputs an imputed time series matrix to fool the discriminator D. Moreover, the classifier C is trained with the labeled time series,

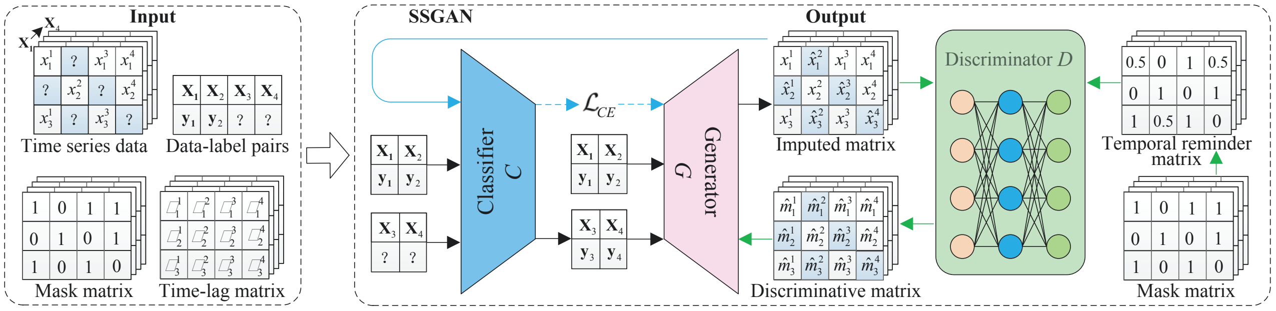


Figure 1: The architecture of SSGAN

which is used to predict the labels for the unlabeled time series. The classifier C gives the feedback, i.e., the cross entropy loss, denoted by $\mathcal{L}_{CE}$, to the generator G. It guides the generator to focus more on the time series samples with the same label when imputing an incomplete sample.

On the other hand, the discriminator D takes the imputed time series matrix and a *temporal reminder matrix* as inputs. It attempts to distinguish the components imputed by the generator from the observed components. In particular, the temporal reminder matrix is introduced to encode a fraction of the missing information in time series data (stored in the mask matrix). The output of the discriminator is a discriminative matrix, containing the probability that presents the authenticity degree of each component.

For the three players in SSGAN, we introduce the BiRNN cell with the *bidirectional* recurrent neural network. The BiRNN cell adopts a time decay vector to effectively capture the important informations of past or future steps in multivariate time series (Cao et al. 2018; Che et al. 2018; Liu et al. 2017). The basic architecture of BiRNN cell is depicted in Figure 2. It takes a given time series sample $\mathbf{X}$, a forward time-lag matrix $\boldsymbol{\delta}$, and a backward time-lag matrix $\boldsymbol{\delta}'$ as inputs. The outputs of BiRNN cell are the reconstructed time series $\bar{\mathbf{X}}$ and $\bar{\mathbf{X}}'$ of both directions. Taking the forward direction (w.r.t. the grey box in Figure 2) as an example, the updating steps of BiRNN can be described by the following.

- To fit the decayed influence of the past observations, the time decay vector, denoted by $\boldsymbol{\gamma}_t$, is modeled as a combination of $\boldsymbol{\delta}_t$ to control the influence of the past observations, i.e., $\boldsymbol{\gamma}_t = \exp\{-\max(0, \mathbf{W}_\gamma \boldsymbol{\delta}_t + \mathbf{b}_\gamma)\}$.
- BiRNN uses a fully connected layer to reconstruct $\bar{\mathbf{x}}_t$ with the previous hidden state $\mathbf{h}_{t-1}$, i.e., $\bar{\mathbf{x}}_t = \mathbf{W}_x \mathbf{h}_{t-1} + \mathbf{b}_x$. Then, it replaces missing values in $\mathbf{x}_t$ with the corresponding values in $\bar{\mathbf{x}}_t$, i.e., $\hat{\mathbf{x}}_t = \mathbf{m}_t \odot \mathbf{x}_t + (1 - \mathbf{m}_t) \odot \bar{\mathbf{x}}_t$.
- Using the time decay vector γ_t and the imputed vector $\hat{\mathbf{x}}_t$, BiRNN calculates the next hidden state $\mathbf{h}_t$ by Eq. 1.

$$\mathbf{h}_t = \sigma\left(\mathbf{W}_h(\boldsymbol{\gamma}_t \odot \boldsymbol{h}_{t-1}) + \mathbf{U}_h \hat{\mathbf{x}}_t + \mathbf{b}_h\right) \tag{1}$$

In particular, $\mathbf{W}_\gamma$, $\mathbf{W}_x$, $\mathbf{W}_h$, $\mathbf{b}_\gamma$, $\mathbf{b}_x$, $\mathbf{b}_h$, and $\mathbf{U}_h$ are model parameters that we need to learn, σ is the sigmoid function, and $\odot$ is the element-wise multiplication. Finally, in the forward direction of BiRNN, it obtains the forward hidden state $\mathbf{h} = (\mathbf{h}_1, \cdots, \mathbf{h}_n)$ and the forward reconstructed time series

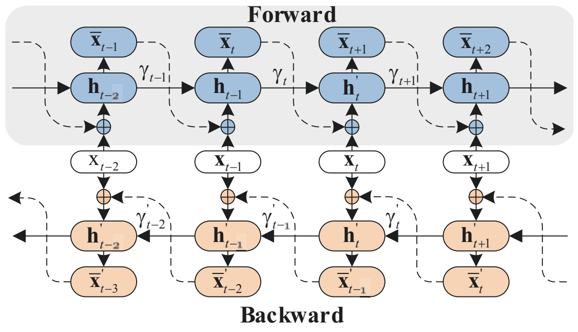


Figure 2: Illustration of the BiRNN cell

matrix $\bar{\mathbf{X}} = (\bar{\mathbf{x}}_1, \cdots, \bar{\mathbf{x}}_n)$. Similarly, in the backward direction, it gets the backward hidden state $\mathbf{h}' = (h'_1, \cdots, h'_n)$ and the backward reconstructed matrix $\bar{\mathbf{X}}' = (\bar{\mathbf{x}}'_1, \cdots, \bar{\mathbf{x}}'_n)$.

SSGAN Details

In this section, we elaborate three players in the SSGAN model. Then, we present a theoretical analysis on the equilibrium of the SSGAN model with the help of the temporal reminder matrix and the classifier.

Generator Network

The purpose of the generator is to learn the distribution of observed multivariate time series data, and then to estimate the missing values in the input time series. That is, the generator takes the matrices $\mathbf{X}$, $\mathbf{M}$, and $\boldsymbol{\delta}$ as inputs, and outputs the imputed time series data $\hat{\mathbf{X}}$. The network structure of the generator is mainly composed of the BiRNN cell.

The objective of the generator contains three types of losses, including the adversarial loss, the classifier loss, and the reconstruction loss. The adversarial loss is the same to the standard GANs. The classifier loss is the feedback from the classifier, which will be defined in Eq. 4. The reconstruction loss is used to enforce the consistency between the observed time series and the reconstructed time series. In particular, we denote the forward and backward reconstructed time series as $\bar{\mathbf{X}}$ and $\bar{\mathbf{X}}'$, respectively. Then, the reconstruction losses $\mathcal{L}(\mathbf{X}, \bar{\mathbf{X}})$ and $\mathcal{L}(\mathbf{X}, \bar{\mathbf{X}}')$ can be formalized as,

$$\mathcal{L}(\mathbf{X}, \bar{\mathbf{X}}) = \mathbf{M} \odot \ell_e(\mathbf{X}, \bar{\mathbf{X}}), \mathcal{L}(\mathbf{X}, \bar{\mathbf{X}}') = \mathbf{M} \odot \ell_e(\mathbf{X}, \bar{\mathbf{X}}')$$

where ℓ_e is the function of the *mean absolute error*. Furthermore, we also calculate the reconstruction error between $\bar{\mathbf{X}}$

and $\bar{\mathbf{X}}'$ to enforce the prediction to be consistent in both directions, i.e., $\mathcal{L}(\bar{\mathbf{X}}, \bar{\mathbf{X}}') = \mathbf{M} \odot \ell_e(\bar{\mathbf{X}}, \bar{\mathbf{X}}')$.

Hence, the overall objective of the generator $\mathcal{L}_G$ can be formalized in Eq. 2.

$$\begin{aligned}\mathcal{L}_G = &\frac{1}{n}(\mathcal{L}(\mathbf{X}, \bar{\mathbf{X}}) + \mathcal{L}(\mathbf{X}, \bar{\mathbf{X}}') + \mathcal{L}(\bar{\mathbf{X}}, \bar{\mathbf{X}}')) \\ &+ \alpha \mathbb{E}[(1-\mathbf{M}) \odot \log(1 - D(\hat{\mathbf{X}}, \mathbf{R}))] + \beta \mathcal{L}_C(y, \hat{\mathbf{X}})\end{aligned} \tag{2}$$

where n denotes the number of timestamps, α and β are hyper-parameters, and $\mathbf{R}$ is the temporal reminder matrix that will be defined by Eq. 3. $\mathbb{E}[(1-\mathbf{M}) \odot \log(1-D(\hat{\mathbf{X}}, \mathbf{R}))]$ is the adversarial loss produced by the fixed discriminator. $\mathcal{L}_C(y, \hat{\mathbf{X}})$ is the classifier loss generated by the fixed classifier. y is the label of the input time series $\mathbf{X}$. $\hat{\mathbf{X}}$ is the output of generator and calculated by $\hat{\mathbf{X}} = \mathbf{M} \odot \mathbf{X} + (1-\mathbf{M}) \odot ((\bar{\mathbf{X}} + \bar{\mathbf{X}}')/2)$. The generator is trained to minimize $\mathcal{L}_G$ using the newly updated discriminator and classifier.

Discriminator Network

Following the standard GAN model, we employ a discriminator to compete with the generator, which can help the generator to generate data as truly as possible. Unlike the standard discriminator whose output is a scalar value, the output of the discriminator in SSGAN is a probability matrix, where each value denotes the authenticity degree of each component in the imputed time series $\hat{\mathbf{X}}$.

The discriminator attempts to distinguish the observed components from the generated ones of the time series $\hat{\mathbf{X}}$ imputed by the generator. Inspired by (Yoon, Jordon, and Schaar 2018), we introduce a *temporal reminder matrix* $\mathbf{R}$ as an additional input to the discriminator. The temporal reminder matrix $\mathbf{R}$ contains a fraction of the missing information in time series. It can be defined as

$$\mathbf{R} = \mathbf{K} \odot \mathbf{M} + 0.5(1 - \mathbf{K}) \tag{3}$$

where the components in $\mathbf{K} = (\mathbf{k}_1, \cdots, \mathbf{k}_i, \cdots, \mathbf{k}_n) \in \{0,1\}^{d \times n}$ are randomly set to 1 or 0 in each training epoch. In other words, $\mathbf{R}$ reveals some components (i.e., their states in $\mathbf{K}$ are set to 1) of $\mathbf{M}$ to the discriminator.

As a result, the discriminator D concatenates the imputed time series $\hat{\mathbf{X}}$ and the temporal reminder matrix $\mathbf{R}$ as the input. The network structure of discriminator is mainly composed of a BiRNN layer and a fully connected layer. The BiRNN in discriminator calculates the next hidden state $\mathbf{h}_t$ by Eq. 1. Then, the fully connected layer is used to output the probability that each component of time series is observed. The forward discriminative probability, denoted by $\bar{\mathbf{m}}_t$, is calculated by $\bar{\mathbf{m}}_t = \mathbf{W}_x^D \mathbf{h}_t^D + \mathbf{b}_x^D$, where $\mathbf{W}_x^D$ and $\mathbf{b}_x^D$ are parameters in the discriminator that we need to learn. Thus, the discriminator obtains a forward discriminative matrix, stored in $\bar{\mathbf{M}} = (\bar{\mathbf{m}}_1, \cdots, \bar{\mathbf{m}}_i, \cdots, \bar{\mathbf{m}}_n) \in \mathbb{R}^{d \times n}$. We can similarly get a backward discriminative matrix $\bar{\mathbf{M}}' = (\bar{\mathbf{m}}'_1, \cdots, \bar{\mathbf{m}}'_i, \cdots, \bar{\mathbf{m}}'_n) \in \mathbb{R}^{d \times n}$. Therefore, the output of the discriminator, denoted by $D(\hat{\mathbf{X}}, \mathbf{R})$, can be written as

$$D(\hat{\mathbf{X}}, \mathbf{R}) = \hat{\mathbf{M}} = \frac{\bar{\mathbf{M}} + \bar{\mathbf{M}}'}{2} = (\hat{\mathbf{m}}_1, \cdots, \hat{\mathbf{m}}_i, \cdots, \hat{\mathbf{m}}_n)$$

In the discriminator, the loss function $\mathcal{L}_D$ is the probability of correctly predicting the mask matrix $\mathbf{M}$, i.e.,

$$\mathcal{L}_D = -\mathbb{E}[\mathbf{M} \odot \log \hat{\mathbf{M}} + (1 - \mathbf{M}) \odot \log(1 - \hat{\mathbf{M}})]$$

The discriminator D is trained by minimizing $\mathcal{L}_D$.

In addition, following the standard GAN, we could directly propose an *unsupervised* generative adversarial network (USGAN for short), which only consists of a generator and a discriminator. Hence, the generator in USGAN has no feedback from the classifier. The generator and discriminator in USGAN are trained simultaneously in a game.

Classifier Network

To generate data with the *partially* labeled data, the traditional GAN models (Nguyen et al. 2017; Odena, Olah, and Shlens 2017) require the discriminator to simultaneously play two roles of identifying generated data and predicting labels. However, the learning capacity of the two-player GANs is limited in semi-supervised learning (Li et al. 2017). Specifically, this discriminator theoretically has two incompatible convergence points. These two roles of the discriminator compete by treating the input data differently. The discriminator pays more attention to the task of identifying generated data, since the task of identifying generated data is much easier than the label prediction task.

To this end, we propose to leverage a semi-supervised classifier in SSGAN to consider the times series imputation and downstream analysis at the same time. Similar as the discriminator, the classifier also consists of a BiRNN layer and a fully connected layer. To minimize model parameters and speed up training, we share the BiRNN layer parameters between the discriminator and classifier, like (Chen et al. 2016; Gong et al. 2019). The classifier adopts the self-training technique, so as to make full use of label information in the partially labeled time series data. Specifically, during training SSGAN, the classifier predicts the labels for a fraction of *unlabeled* time series samples with largest predicted confidences. It then trains with the labeled time series data imputed by the generator. It drives the generator to pay more attention to the time series sharing the same label l, when it is imputing an incomplete time series with a label l.

We try to find a set of network parameters for the classifier, so as to produce a high confidence on the real label for the input time series. The output of the classifier $C(\mathbf{X})$ can be written as

$$C(\mathbf{X}) = \frac{1}{2}(f(\frac{1}{n}\sum_{i=1}^{n} \mathbf{h}_i) + f(\frac{1}{n}\sum_{i=1}^{n} \mathbf{h}'_i))$$

Therefore, the loss function of the classifier $\mathcal{L}_C$ is defined as

$$\mathcal{L}_C(y, \mathbf{X}) = \mathcal{L}_{CE}(y, C(\mathbf{X})) \tag{4}$$

where $\mathcal{L}_{CE}$ indicates the cross-entropy loss function, f is the softmax function, and $\mathbf{h}_i$ and $\mathbf{h}'_i$ are the i-th hidden state of the forward and backward directions, respectively.

Accordingly, SSGAN explicitly assigns a discriminator and a classifier for identifying generated data and predicting labels, respectively. When updating the generator, it receives feedback from the classifier. Thus, in the generator, imputed values in an incomplete time series sample are more likely to be reasoned with the samples with the same label to this incomplete sample. Meanwhile, the classifier is trained via using the time series imputed by the generator. The generator and classifier are optimized simultaneously. Therefore, if one of them tends to convergence, the other one will also converge. As a result, with the help of the classifier, SSGAN effectively addresses the two-role competing problem, and thereby makes the missing data imputation more accurate.

Theoretical Analysis

Let $\hat{\mathbf{X}}$ be a time series sample imputed by the generator G, $\mathbf{R}$ be the corresponding temporal reminder matrix provided to the discriminator D, $D(\hat{\mathbf{X}}, \mathbf{R})$ be the output of D, $C(\hat{\mathbf{X}})$ be the output of the classifier C, and $\mathbf{M}$ be the mask matrix of the input time series data $\mathbf{X}$. $\mathcal{L}_{CE}$ is the cross-entropy loss function. y is the true label of $\hat{\mathbf{X}}$. SSGAN can be defined as a minimax game as follows.

$$\begin{aligned}\min_{G,C}\max_{D} U(G,D,C) &= \mathbb{E}_{\hat{\mathbf{X}},\mathbf{M},\mathbf{R}}[\mathbf{M}\odot\log D(\hat{\mathbf{X}},\mathbf{R}) \\ &\quad + (1-\mathbf{M})\odot\log(1-D(\hat{\mathbf{X}},\mathbf{R}))] + \beta\mathcal{L}_{CE}(y, C(\hat{\mathbf{X}})) \\ &= V(G,D) + \beta L(G,C)\end{aligned}$$

where $\odot$ is the element-wise multiplication and β is a hyperparameter. In particular, the SSGAN model $U(G,D,C)$ can be regarded as the USGAN model $V(G,D)$ with a classifier interacted with the generator, i.e., $L(G,C)$. The distribution of the random variable $(\hat{\mathbf{X}}, \mathbf{M}, \mathbf{R})$ can be defined as $p(\mathbf{X}, \mathbf{M}, \mathbf{R})$, and the marginal distributions of $\hat{\mathbf{X}}$, $\mathbf{M}$, and $\mathbf{R}$ can be defined as $\hat{p}$, p_m, and p_r, respectively.

Based on the minimax game theory, we can conclude that, using the temporal reminder matrix and the classifier ensures that the distributions produced by SSGAN converge to the true data and label distributions, when the Nash equilibrium is achieved. We first prove that, the equilibrium of the USGAN model $V(G,D)$ can be uniquely determined with the help of the temporal reminder matrix in Theorem 1. Then, we confirm that, with the optimization of $V(G,D)$, the SSGAN model $U(G,D,C)$ converges to the Nash equilibrium, and the distribution produced by G or C in SSGAN is the same as the true data or label distribution in Theorem 2. In particular, the following theorems are under the assumption that, $\mathbf{M}$ is independent of $\mathbf{X}$, i.e., the time series data are missing completely at random (i.e., MCAR) (Rubin 1976).

Theorem 1 *Given a temporal reminder matrix defined in Eq. 3, the equilibrium of the USGAN model $V(G,D)$ can be uniquely determined. The distribution $\hat{p}(\mathbf{X}|\mathbf{M})$ produced by the USGAN model is the same as the true data distribution $\hat{p}(\mathbf{X}|\mathbf{1})$. Namely, $\hat{p}(\mathbf{X}|\mathbf{M}) = \hat{p}(\mathbf{X}|\mathbf{1})$.*

The proof of Theorem 1 completes based on the work (Yoon, Jordon, and Schaar 2018), and the proof details are described in Appendix A.

Theorem 2 *The Nash equilibrium of the SSGAN model $U(G,D,C)$ is achieved if and only if $\hat{p}(\mathbf{X}|\mathbf{M}) = \hat{p}(\mathbf{X}|\mathbf{1})$ and $p_c(\cdot|\mathbf{X}) = p_l(\cdot|\mathbf{X})$, where $p_c(\cdot|\mathbf{X})$ is the distribution of predicting label conditional on $\mathbf{X}$, and $p_l(\cdot|\mathbf{X})$ is the distribution of true label conditional on $\mathbf{X}$.*

Proof. In the definition of the SSGAN model $U(G,D,C) = V(G,D) + \beta L(G,C)$, we have $L(G,C) = \mathcal{L}_{CE}(y, C(\hat{\mathbf{X}}))$. It can be rewritten as

$$\begin{aligned}L(G,C) =& \mathcal{L}_{CE}(y, C(\hat{\mathbf{X}})) \\ =& \mathbb{E}_{\hat{\mathbf{X}}\sim G(\mathbf{X})}[\mathbb{E}_{y'\sim p_l(y|\hat{\mathbf{X}})}[-\log p_c(y'|\hat{\mathbf{X}})]] \\ =& \mathbb{E}_{\hat{\mathbf{X}}\sim G(\mathbf{X})}[D_{KL}(p_l(\cdot|\hat{\mathbf{X}})||p_c(\cdot|\hat{\mathbf{X}})) \\ &+ \mathbb{E}_{y'\sim p_l(y|\hat{\mathbf{X}})}[-\log p_l(y'|\hat{\mathbf{X}})]]\end{aligned}$$

where the second equality is from the definition of the cross-entropy loss function, i.e., $\mathcal{L}_{CE}(q,q') = -\sum_x q(x)\log q'(x)$. Namely, for SSGAN, minimizing $L(G,C)$ is equivalent to minimizing the Kullback-Leibler divergence $D_{KL}(p_l(\cdot|\hat{\mathbf{X}})||p_c(\cdot|\hat{\mathbf{X}}))$, which is non-negative and zero if and only if $p_l(\cdot|\hat{\mathbf{X}}) = p_c(\cdot|\hat{\mathbf{X}})$. Meanwhile, applying Theorem 1 to $U(G,D,C)$, we have $\hat{p}(\mathbf{X}|\mathbf{M}) = \hat{p}(\mathbf{X}|\mathbf{1})$ at the equilibrium of SSGAN. Therefore, the distribution produced by G or C in SSGAN is the same as the true data distribution (i.e., $\hat{p}(\mathbf{X}|\mathbf{M}) = \hat{p}(\mathbf{X}|\mathbf{1})$) or label distribution (i.e., $p_l(\cdot|\hat{\mathbf{X}}) = p_c(\cdot|\hat{\mathbf{X}})$) at the equilibrium. □

Experimental Evaluation

In this section, we evaluate the performance of our proposed SSGAN model via comparisons with nine state-of-the-art methods. All methods were implemented in Python. The experiments were conducted in an Intel Core 2.80GHz server with TITAN Xp 12GiB (GPU) and 192GB RAM.

In the experiments, we use three benchmark multivariate time series datasets: (i) The localization for human activity dataset[1] (**Activity**) (Cao et al. 2018; Kaluža et al. 2010; Silva et al. 2012) consists of the multivariate kinematic time series recording the motion state of 5 people performing 11 kinds of activities. Each person wore four sensors (tags) on her/his left/right ankle, chest, and belt to record the 3-dimensional coordinates. There are 4,100 time series samples used in the experiments over 40 consecutive time steps. (ii) The PhysioNet Challenge 2012 dataset[2] (**PhysioNet**) (Cao et al. 2018; Che et al. 2018; Luo et al. 2018, 2019; Silva et al. 2012) contains 4,000 multivariate clinical time series with 41 measurements from intensive care unit (ICU) stays, where 554 patients died in hospital. Each sample is captured at the first 48 hours after the patient admission to ICU. There are 80.67% missing values in the original dataset. (iii) The KDD CUP 2018 dataset[3] (**KDD**) (Cao et al. 2018; Luo et al. 2018, 2019; Silva et al. 2012), a public meteorologic dataset with 13.30% missing rate, includes PM2.5 measurements from 36 monitoring stations in Beijing, which are hourly collected between 2014/05/01 to 2015/04/30. There are totally 365 time series samples with a same label in 24 consecutive hours.

In the experiments, we utilize *root mean square error* (RMSE) (Jeffery, Garofalakis, and Franklin 2006) and *mean absolute error* (MAE) (Chai and Draxler 2014) to measure the imputation performance. For both of them, the smaller

[1]https://archive.ics.uci.edu/ml/datasets/Localization+Data+for+Person+Activity

[2]https://physionet.org/content/challenge-2012/1.0.0/

[3]https://github.com/NIPS-BRITS/BRITS

Dataset	Missing	Zero	Mean	Last	GRUD	GP-VAE	TSGAN	E^2GAN	NAOMI	BRITS	SSGAN
Activity	10%	0.813	0.813	0.792	0.722	0.670	0.705	0.677	0.641	0.621	**0.600**
	30%	0.863	0.873	0.862	0.792	0.726	0.785	0.762	0.724	0.686	**0.666**
	50%	0.929	0.933	0.936	0.879	0.796	0.825	0.812	0.794	0.786	**0.759**
	70%	0.949	0.943	0.956	0.923	0.846	0.845	0.842	0.854	0.836	**0.803**
	90%	0.957	0.963	0.968	0.951	0.882	0.859	0.876	0.897	0.867	**0.841**
PhysioNet	10%	0.805	0.799	0.802	0.710	0.677	0.691	0.679	0.632	0.611	**0.598**
	30%	0.853	0.863	0.855	0.782	0.707	0.782	0.739	0.703	0.672	**0.670**
	50%	0.922	0.916	0.917	0.808	0.787	0.812	0.789	0.783	0.779	**0.762**
	70%	0.956	0.936	0.947	0.848	0.837	0.829	0.823	0.835	0.809	**0.782**
	90%	0.963	0.952	0.959	0.863	0.879	0.847	0.853	0.865	0.850	**0.818**
KDD	10%	0.781	0.763	0.789	0.701	0.522	0.682	0.654	0.522	0.531	**0.435**
	30%	0.812	0.806	0.804	0.761	0.562	0.752	0.675	0.558	0.561	**0.461**
	50%	0.892	0.866	0.888	0.801	0.602	0.783	0.725	0.602	0.581	**0.490**
	70%	0.926	0.898	0.921	0.821	0.709	0.796	0.768	0.701	0.641	**0.603**
	90%	0.968	0.919	0.949	0.842	0.771	0.829	0.790	0.762	0.720	**0.660**

Table 1: Performance comparison (in RMSE) of time series imputation methods under different missing rates

the value, the better the imputation effect. In order to obtain the metric values, we randomly remove 50% observed values under MCAR during training for the imputation task, and we use them as the ground-truth to calculate the RMSE and MAE values. Due to the similar trends of RMSE and MAE and the space constraint, MAE-based experimental results are presented in Appendix B. In each set of experiments, the average value of five times experimental results is reported, under random divisions of training and test data.

For Activity and PhysioNet, we do the imputation task. For PhysioNet, we also perform the post-imputation classification task. We randomly choose 80% data as the training data, and the rest as the test data. For KDD, we do the imputation and post-imputation regression tasks. Following the prior work (Yi et al. 2015), we use the 3-rd, 6-th, 9-th, and 12-th months as the test data and the other as the training data. We select every 24 consecutive hours as one time series to predict the mean air quality of the next 6 hours.

We compare our proposed SSGAN model with nine existing state-of-the-art time series imputation methods, including Zero imputation, Mean imputation, Last imputation, GRUD (Che et al. 2018), GP-VAE (Fortuin et al. 2019), TSGAN (Luo et al. 2018), E^2GAN (Luo et al. 2019), NAOMI (Liu et al. 2019), and BRITS (Cao et al. 2018).

In implementation, for all deep learning baselines, the learning rate is 0.001, the dropout rate is 0.5, the number of hidden units in recurrent network is 64, and the training epoch is 30. The batch size is 128 for PhysioNet and 64 for KDD and Activity. The dimensionality of the latent space in GP-VAE is 35. The dimensionality of random noise in TSGAN and the latent vector in E^2GAN are both 64. In SSGAN, the ADAM algorithm is utilized to train networks, the classifier is pre-trained for five epochs before training GAN model, the training epoch is 30, the generator is trained once for every five optimization steps of the discriminator. The hyper-parameters α and β of the generator are both 5. The reminder rate (i.e., how many missing states in $\mathbf{M}$ are encoded in the temporal reminder matrix for the discriminator) is 0.8. The label rate is 100%. Unless mentioned otherwise, the parameters get their default values in the experiments.

Comparison Study

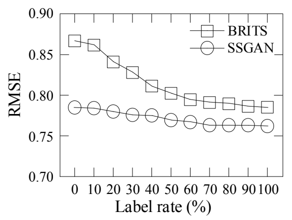
The first set of experiments explores the performance of different imputation methods on the imputation task when the missing rate is 50% by default. The corresponding imputation results are included in Table .

One can observe that, SSGAN outperforms all baselines. Specifically, SSGAN exceeds the best performing state-of-the-art baseline (i.e., BRITS) by 6.29% in average, and even increases up to 15.67% over BRITS on KDD. This is because, SSGAN adopts the semi-supervised GAN model with the temporal reminder matrix to learn the true data distribution, and thereby enhances the imputation accuracy. To be more specific, taking the state-of-the-art unsupervised imputation method NAOMI as a baseline, the performance increase of SSGAN is 4.41x, 5.25x, and 5.33x as much as that of BRITS on Activity, PhysioNet, and KDD, respectively. In general, statistical methods perform worse than deep learning methods. Among deep learning methods, GRUD is not very effective.

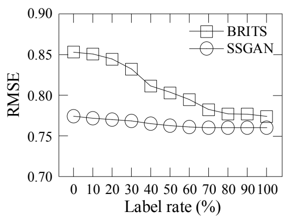
Parameter Evaluation

Effect of missing rate. When varying the missing rate (i.e., how many values are missing in observed time series data for test) from 10% to 90%, the corresponding results are listed in Table over three datasets. We can find that, SSGAN also gets the best performance in each case. Furthermore, with the growth of missing rate, the imputation accuracy descends consistently for each algorithm. The reason is that, as the missing rate increases, the data information for imputation algorithms becomes less, and thereby the imputation algorithms perform less effective.

Effect of label rate. We inspect the effect of the label rate (i.e., how many labels of time series are used in experiments) on the performance of the semi-supervised (i.e., SSGAN) and supervised (i.e., BRITS) imputation methods over Activity and PhysioNet. The experimental results of varying the label rate from 0 to 100% are plotted in Figure 3, where the labels are missing randomly. One can observe that, SSGAN and BRITS perform better with the increasing label rate. It is obvious that, when there is only a small fraction of time



(a) Activity



(b) PhysioNet

Figure 3: Comparison of BRITS and SSGAN on label rates

Dataset	SSGAN	USGAN	S-no-D	S-no-R
Activity	0.759	0.772 (1.68%)	**0.779 (2.57%)**	0.773 (1.81%)
PhysioNet	0.762	0.770 (1.04%)	**0.771 (1.17%)**	0.766 (0.52%)

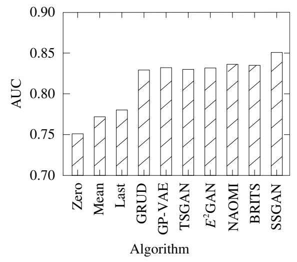
Table 2: The ablation study on SSGAN (Gain(%))

series having labels (w.r.t. a small label rate), SSGAN exhibits the significant advantages, compared to BRITS. This is partially because that, the classifier in SSGAN constantly predicts the labels for unlabeled samples, making SSGAN obtain more labeled training data. However, for BRITS, the performance turns much worse for the times series with less labels. The reason is that, the small label rate indicates the less number of labeled samples used to train BRITS, resulting in poor performance of BRITS. Thus, SSGAN is much more effective than BRITS, especially for real-life time series data with partial labels.

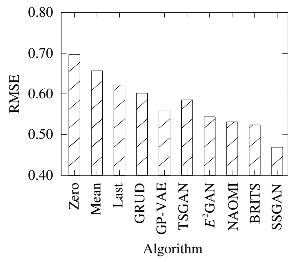
In addition, we study the effect of the *reminder rate* (i.e., how much missing information is encoded in the reminder matrix $\mathbf{R}$) and the *iteration times* of discriminator per iteration on the SSGAN performance, respectively. The detailed experimental results are described in Appendix C. One can observe that, SSGAN basically achieves the best performance when the reminder rate is 80% and the iteration times of discriminator is 5. Moreover, when the reminder matrix $\mathbf{R}$ is highly close to the mask matrix $\mathbf{M}$ (i.e., the reminder rate approximates to 1), SSGAN becomes worse. Thus, it further confirms that, the reminder matrix $\mathbf{R}$ (instead of $\mathbf{M}$) is indeed beneficial to SSGAN.

Ablation Study

We investigate the influence of different elements of SSGAN on the imputation performance. The corresponding experimental results are shown in Table . USGAN is actually the variant of SSGAN without classifier. S-no-D (resp. S-no-R) denotes the variant of SSGAN with no discriminator (resp. no temporal reminder matrix). We can observe that, each element, i.e., the classifier, the discriminator, and the temporal reminder matrix, in SSGAN does have a positive effect on the imputation performance. In particular, the imputation accuracy decreases in average 1.36%, 1.87%, and 1.17% without the classifier, discriminator, or temporal reminder matrix, respectively. The discriminator in SSGAN has the largest effect. It confirms that, adversarial training strategy does boost performance. Meanwhile, fully using the time series data with some labels is indeed beneficial for imputation, as well as using the temporal reminder matrix.



(a) Classification on PhysioNet



(b) Regression on KDD

Figure 4: Post-imputation evaluation on imputation methods

Post-imputation Prediction

We study the performance of those imputation algorithms on post-imputation prediction, which indirectly reflects the performance of each imputation method. The post-imputation prediction results are depicted in Figure 4, with the classification task over PhysioNet and the regression task over KDD. The larger AUC value corresponds to the better post-imputation prediction effect, while RMSE is opposite. In particular, the imputation methods are first employed to impute missing values in original incomplete datasets which have *complex real* missing values. Then, we use a simple RNN layer to construct the RNN classifier and RNN regression models, and we train the models with imputed datasets. The training epoch is 30, the learning rate is 0.005, the pre-training epoch is 5, the dropout rate is 0.5, and the dimensionality of the hidden states in RNN is 64.

First, we can find that, these results are consistent with those in imputation task, i.e., SSGAN outperforms the others, and deep learning imputation methods perform much better than statistical imputation methods. Specifically, in the post-imputation prediction, SSGAN exceeds the best performing state-of-the-art method in baseline (i.e., BRITS) by 9.24% in average. It even increases up to 17.20% over BRITS on KDD. In terms of one imputation method applied to different prediction tasks, we can conclude that, the improvement of SSGAN on the regression task over KDD is bigger than that on the classification task over PhysioNet. In addition, we can further observe that, supervised and semi-supervised methods get higher prediction accuracy than unsupervised ones in most cases. GRUD performs well on the prediction of PhysioNet, while it has limitations on KDD.

Conclusions

In this paper, we propose a novel semi-supervised generative adversarial network model SSGAN, with a generator, a discriminator, and a classifier, for multivariate time series data imputation. We introduce a temporal reminder matrix to assist the discriminator to better distinguish the observed components from the ones imputed by the generator. We present a semi-supervised classifier to solve the insufficient label issue. Moreover, it confirms that, SSGAN has the unique equilibrium with the help of the temporal reminder matrix and the semi-supervised classifier. Extensive experiments using benchmark time series datasets demonstrate that, SSGAN significantly boosts the imputation and prediction performance, compared with the state-of-the-art methods.

Acknowledgments

This work is partly supported by the National Key Reserch and Development Program of China under Grant No.2017YFB1400601, the NSFC under Grants No.61902343, No.62025206, No.61972338, No.61825205, No.61772459, and the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under Grant No.LR21F020005. Jianwei Yin is the corresponding author of this work.

References

Amiri, M.; and Jensen, R. 2016. Missing data imputation using fuzzy-rough methods. *Neurocomputing* 205(1): 152–164.

Bai, J.; and Ng, S. 2008. Forecasting economic time series using targeted predictors. *Journal of Econometrics* 146(2): 304–317.

Bartholomew, D. J. 1971. Time series analysis forecasting and control. *Journal of the Operational Research Society* 22(2): 199–201.

Bright, J.; Smith, C.; Taylor, P.; and Crook, R. 2015. Stochastic generation of synthetic minutely irradiance time series derived from mean hourly weather observation data. *Solar Energy* 115(1): 229–242.

Cao, W.; Wang, D.; Li, J.; Zhou, H.; Li, L.; and Li, Y. 2018. BRITS: Bidirectional recurrent imputation for time series. In *NeurIPS*, 6775–6785.

Chai, T.; and Draxler, R. R. 2014. Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. *Geoscientific Model Development* 7(3): 1247–1250.

Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; and Liu, Y. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports* 8(1): 6085.

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, 2172–2180.

Esteban, C.; Hyland, S. L.; and Rätsch, G. 2017. Real-valued (medical) time series generation with recurrent conditional gans. *ArXiv Preprint ArXiv:1706.02633* .

Fortuin, V.; Baranchuk, D.; Rätsch, G.; and Mandt, S. 2019. GP-VAE: Deep Probabilistic Time Series Imputation. *ArXiv Preprint ArXiv:1907.04155* .

Goldberger, A. L.; Amaral, L. A.; Glass, L.; Hausdorff, J. M.; Ivanov, P. C.; Mark, R. G.; Mietus, J. E.; Moody, G. B.; Peng, C.-K.; and Stanley, H. E. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation* 101(23): 215–220.

Gong, X.; Chang, S.; Jiang, Y.; and Wang, Z. 2019. Autogan: Neural architecture search for generative adversarial networks. In *ICCV*, 3224–3234.

Hamzaçebi, C. 2008. Improving artificial neural networks' performance in seasonal time series forecasting. *Information Sciences* 178(23): 4550–4559.

Jeffery, S. R.; Garofalakis, M.; and Franklin, M. J. 2006. Adaptive cleaning for RFID data streams. In *VLDB*, 163–174.

Kaluža, B.; Mirchevska, V.; Dovgan, E.; Luštrek, M.; and Gams, M. 2010. An agent-based approach to care in independent living. In *IJCAI*, 177–186.

Li, C.; Xu, T.; Zhu, J.; and Zhang, B. 2017. Triple generative adversarial nets. In *NeurIPS*, 4088–4098.

Liu, Y.; Fu, J.; Mei, T.; and Chen, C. W. 2017. Let your photos talk: Generating narrative paragraph for photo stream via bidirectional attention recurrent neural networks. In *AAAI*, 1445–1452.

Liu, Y.; Yu, R.; Zheng, S.; Zhan, E.; and Yue, Y. 2019. NAOMI: Non-Autoregressive Multiresolution Sequence Imputation. In *NeurIPS*, 11236–11246.

Luo, Y.; Cai, X.; Zhang, Y.; and Xu, J. 2018. Multivariate time series imputation with generative adversarial networks. In *NeurIPS*, 1596–1607.

Luo, Y.; Zhang, Y.; Cai, X.; and Yuan, X. 2019. E^2GAN: end-to-end generative adversarial network for multivariate time series imputation. In *IJCAI*, 3094–3100.

Ma, J.; Shou, Z.; Zareian, A.; Mansour, H.; Vetro, A.; and Chang, S.-F. 2019. CDSA: Cross-Dimensional Self-Attention for Multivariate, Geo-tagged Time Series Imputation. *ArXiv Preprint ArXiv:1905.09904* .

Mattei, P. A.; and Frellsen, J. 2019. MIWAE: Deep generative modelling and imputation of incomplete data sets. In *ICML*, 4413–4423.

Miao, X.; Gao, Y.; Chen, G.; Zheng, B.; and Cui, H. 2016. Processing incomplete k nearest neighbor search. *IEEE Transactions on Fuzzy Systems* 24(6): 1349–1363.

Miao, X.; Gao, Y.; Guo, S.; Chen, L.; and Li, Q. 2019. Answering Skyline Queries over Incomplete Data with Crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering* 10.1109/TKDE.2019.2946798.

Nguyen, A.; Clune, J.; Bengio, Y.; Dosovitskiy, A.; and Yosinski, J. 2017. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, 4467–4477.

Odena, A.; Olah, C.; and Shlens, J. 2017. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2642–2651.

Rubin, D. B. 1976. Inference and missing data. *Biometrika* 63(3): 581–592.

Shi, Q.; Yin, J.; Cai, J.; Cichocki, A.; Yokota, T.; Chen, L.; Yuan, M.; and Zeng, J. 2020. Block Hankel Tensor ARIMA for Multiple Short Time Series Forecasting. In *AAAI*, 5758–5766.

Silva, I.; Moody, G.; Scott, D. J.; Celi, L. A.; and Mark, R. G. 2012. Predicting in-hospital mortality of icu patients:

The physionet/computing in cardiology challenge 2012. In *CinC*, 245–248.

Song, D.; Xia, N.; Cheng, W.; Chen, H.; and Tao, D. 2018. Deep r-th root of rank supervised joint binary embedding for multivariate time series retrieval. In *SIGKDD*, 2229–2238.

Wang, J.; Wang, Z.; Li, J.; and Wu, J. 2018. Multilevel wavelet decomposition network for interpretable time series analysis. In *SIGKDD*, 2437–2446.

Yi, X.; Zheng, Y.; Zhang, J.; and Li, T. 2015. ST-MVL: Filling missing values in geo-sensory time series data. In *IJCAI*, 2704–2710.

Yoon, J.; Jordon, J.; and Schaar, M. 2018. GAIN: Missing data imputation using generative adversarial nets. In *ICML*, 5675–5684.

Yoon, J.; Zame, W. R.; and van der Schaar, M. 2017. Multi-directional recurrent neural networks: A novel method for estimating missing data. In *ICML*, 1–5.