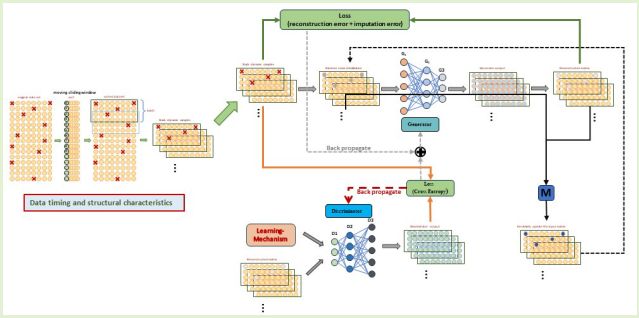
Missing Data Imputation for Industrial Time Series With Adaptive Median Iteration Based on Generative Adversarial Networks

Xiaofeng Yuan, *Senior Member, IEEE*, Jiale Zhang, Kai Wang, *Member, IEEE*, Yalin Wang, *Senior Member, IEEE*, Chunhua Yang, *Fellow, IEEE*, Weihua Gui, *Member, IEEE*, Feifan Shen, *Member, IEEE*, and Lingjian Ye, *Member, IEEE*

***Abstract*—Time series in industrial processes often exhibits missing data caused by inevitable factors such as equipment failures and sensor errors. These missing data include vital information for the production process and directly impact subsequent modeling and analysis. Traditional imputation methods usually face challenges in capturing complex data distributions, structures, and time-dependent relationships. To address this issue, this article proposes an innovative generative adversarial network (GAN)-based stacked adaptive median iterative imputation framework, named as SAMIIF. The model employs techniques such as moving windows and dynamic sample overlay to impute missing time series data in industrial production processes through mean recursive updates, enhancing the understanding of temporal data. In addition, an adaptive learning mechanism framework is introduced to dynamically provide learning information to the discriminator during training. Finally, extensive experiments are conducted to validate the superior performance of the proposed methods on real industrial datasets.**



***Index Terms*—Generative adversarial network (GAN), industrial time series, stacked adaptive mean iteration imputation framework (SAMIIF), time series data loss.**

I. Introduction

TIME series data are valuable resource with extensive applications in industrial production [1], [2]. Through real-time monitoring, recording, and in-depth analysis of time series data, industrial enterprises can effectively enhance production efficiency, reduce costs, and elevate product quality, thereby maintaining competitiveness and achieving sustainable development goals [3], [4]. However, there are many inevitable factors such as equipment failures, sensor errors, network communication issues, human errors, and natural disasters that may lead to intermittent data collection or the loss of partial information [5], [6], [7]. These incomplete data in the time series easily cause bias in data analysis, diminish the reliability of decision-making, and delay issue recognition. The absence of data may result in the inability of the model to accurately capture dynamic changes in the process, thereby affecting prediction accuracy and impacting the performance of the data-driven models for quality prediction [8], [9], [10]. This, in turn, directly influences the precision of downstream models and may even lead to modeling failure [11], [12]. Hence, the correct handling of missing data and the precise completion of absent data in industrial time series constitute a challenging and pressing task, crucial for ensuring the accuracy of analytical results and the reliability of decisions.

In general, based on the factors for missing data and their relationship with observed data, missing data can be categorized into three types: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR) [13]. In the case of MCAR, the missing data are a purely random event, whereas both MAR and MNAR depend

Received 29 July 2024; accepted 5 September 2024. Date of publication 16 September 2024; date of current version 31 October 2024. This work was supported in part by the Program of National Natural Science Foundation of China under Grant 61988101, Grant 62173346, Grant 92267205, and Grant 62303494; in part by the Program of Foundation of Hunan, China under Grant 2021RC4054; and in part by the Fundamental Research Funds for the Central Universities of Central South University under Grant 2024ZZTS0849. The associate editor coordinating the review of this article and approving it for publication was Dr. Vinay Chakravarthi Gogineni. *(Corresponding author: Xiaofeng Yuan.)*

Xiaofeng Yuan, Jiale Zhang, Kai Wang, Yalin Wang, Chunhua Yang, and Weihua Gui are with the School of Automation, Central South University, Changsha 410083, China (e-mail: yuanxf@csu.edu.cn; zjl123jia@163.com; kaiwang@csu.edu.cn; ylwang@csu.edu.cn; ychh@csu.edu.cn; gwh@csu.edu.cn).

Feifan Shen is with the School of Information Science and Engineering, NingboTech University, Ningbo 315100, China (e-mail: ffshen@nbt.edu.cn).

Lingjian Ye is with the School of Engineering, Huzhou University, Huzhou 313000, China (e-mail: lingjian.ye@gmail.com).

Digital Object Identifier 10.1109/JSEN.2024.3457625

on the observed variables, with MNAR additionally influenced by unobserved variables [14]. This study specifically focuses on the issue of MCAR-type missing data of time series in industrial processes, arising from unforeseeable factors such as sensor errors and equipment malfunctions. The imputation of missing time series data aims to make the most of all available data by leveraging observable information within the incomplete dataset through specialized techniques, thereby avoiding wastage of informational resources [15], [16]. This process is crucial for ensuring data integrity and the accuracy of analytical results, especially in industries heavily reliant on data.

In industrial processes, due to the difficulty in obtaining complete high-quality datasets, the common practice is to employ direct deletion methods to handle incomplete data [17], [18]. However, this approach may lead to the loss of crucial information, thereby impacting the accuracy of the model [19]. Technological advancements have spurred the demand for more precise methods to fill missing data, resulting in three primary approaches: statistical analysis, machine learning, and deep learning [20].

Statistical analysis methods, such as mean imputation and median filling, utilize overall mean or median values to estimate missing values. These methods are simple and quick, preserving the characteristics of the original data distribution, but they overlook the relationships between missing data and other variables [21], [22]. On the other hand, machine learning-based methods, such as K-nearest neighbors (KNNs) [23], [24], matrix factorization (MF) [25], and iterative singular value decomposition (iterative SVD) [26], process data through mathematical features and statistical principles. They are more widely applicable and capable of handling diverse data types but require training on complete datasets [27], [28].

With the development of deep learning, neural network models can more effectively extract data features, achieving high-precision fitting and addressing challenges faced by traditional machine learning [29], [30]. Recurrent neural network (RNN) [31], [32], [33], as a typical algorithm for handling sequential data, can capture time-dependent relationships [17], [34]. For instance, Cao et al. [35] developed the time series missing value estimation method to effectively handle missing values through bidirectional recurrent dynamic inputs. However, it requires supervised learning and is limited when applied to datasets with missing samples and labels [36]. Du et al. [37] introduced a method for estimating missing values in multivariate time series based on self-attention mechanisms. This method utilizes two diagonally-masked self-attention (DMSA) blocks for weighted combination to learn missing values, capturing both time dependencies and feature correlations. However, in situations with limited data, the model is prone to overfitting issues. Also, semisupervised learning is widely applied in data imputation. This approach can train models using a small amount of labeled data combined with a large amount of unlabeled data, thereby maximizing the use of all available data [38]. For example, Shao et al. [39] introduced a novel semisupervised training framework with optimized time series for LDS (TD-SsLDS) to address the limitations of supervised learning and the neglect of time series, thereby mitigating the issue of time misalignment due to data absence to some extent. In addition, Xiao et al. [40] proposed a distributed semisupervised HMM (DisSsHMM), which divides data into continuous data blocks (DBs) to enable effective segmental computation in the face of discontinuous data, thus alleviating the impact of data chain interruptions. However, semisupervised learning is more often used for labeled data prediction, which still has limitations in handling high-dimensional data and learning complex distributions.

In recent years, the prominent generative adversarial network (GAN) models have been characterized as unsupervised learning problems [41]. The generative adversarial imputation net (GAIN) model proposed by Yoon et al. [13] introduces the GAN into the task of imputing missing data, incorporating a hint mechanism to make the mapping distribution generated by the generator closer to the distribution of real data, thereby achieving significant completion effects [29], [42]. Miao et al. [43] proposed a novel semisupervised GAN (SSGAN) model, which improves the imputation of missing values in multivariate time series data by introducing a classifier and a time reminder matrix. Luo et al. [44] combined an improved gated recurrent unit with GAN to propose GAN-2-Stage. Furthermore, Zhang et al. [1] developed an end-to-end model to impute missing values in multivariate time series, addressing the input optimization stage issues present in the GAN-2-Stage model. However, during the early stages of GAN model training, the ability of the generator to generate samples and the discriminator to discern real from fake samples are initially limited. Therefore, the discriminator requires minimal hint information about real samples, and excessive hints may lead to several unexpected issues. First, the generator struggles to produce high-quality samples, resulting in a decline in the quality of generated samples and even potential termination of the generator learning, leading to model degradation. Second, the generator may attempt to evade the discriminator, generating similar samples, thus compromising diversity and causing mode collapse. Finally, due to the GAN model being in a min-max game, training difficulties arise in achieving convergence. Excessive hint information given to the discriminator in the early stages of training makes it challenging for the generator to obtain useful gradients, increasing training complexity and time costs.

As the model undergoes further training, the generator's ability to generate samples gradually improves. Therefore, the discriminator requires more hints about real samples to enhance its discriminative ability. Otherwise, the discriminator will be unable to provide useful gradient signals to update the generator, resulting in difficulties in assessing sample authenticity. The generator may also overfit the training data, leading to a decrease in model accuracy, increase the training time and training instability. In addition, these algorithms mostly do not consider the temporal dependency of the data in the time series.

To address these issues, this article proposes an innovative unsupervised learning framework called the stacked adaptive mean iterative imputation framework (SAMIIF). This framework, based on GANs, first utilizes a moving window to

obtain continuous subsequences as model inputs, preserving the temporal information of the data, enabling the model to learn dependencies between sequences. Subsequently, within each sub-block, a dynamic sample overlay training method is employed to increase the model training depth and enhance its understanding of time series data. Finally, by using the output of a sub-block and the mean of its input as the input for the next sub-block, a mean recursive update is implemented to smooth the model predictions, allowing imputation information to propagate during training, thereby improving input efficiency and accuracy. In addition, an adaptive learning mechanism module is designed to dynamically provide learning information for the discriminator during training, preventing issues such as model degradation, mode collapse, and convergence difficulties resulting from imbalances in the training of the generator and discriminator. In summary, the main contributions of this article are as follows.

1) Using recursive averaging of inputs for the reconstruction and imputation of missing values ensures that imputation information is effectively propagated during the training process.
2) Stacked adaptive mean iteration imputation framework based on GAN is designed with moving windows and dynamic sample stacking techniques. The aim is to preserve temporal information in the data and enhance the model's understanding of time series data.
3) A novel adaptive learning mechanism module is proposed to dynamically provide the discriminator with adaptive learning information during the training process, thereby enhancing the stability and training effectiveness of the model.

The remaining parts of this article are structured as follows: Section II provides a brief introduction to GAN. Section III details the proposed stacked adaptive mean iteration imputation framework (SAMIIF). In Section IV, experiments are conducted to demonstrate the superiority of the proposed method over other baseline approaches. Finally, Section V presents a concise summary of the main research findings of this article.

II. Preliminaries

A. Problem Formulation

This article primarily investigates the issue of missing data imputation in time series within industrial processes and introduces specific symbols to describe this problem.

A sequence $X = (x_{t_1} \times \ldots \times x_{t_n}) \in R^{n\times q}$ can represent a time series observed in $T = (t_1, t_2, \ldots, t_n)$ with d dimensions. $P(X)$ denotes the distribution of X. x_{t_i} is the observation at t_i, and $x_{t_i}^j$ is the j^{th} feature in x_{t_i}. To represent the missing time series data, we defined the mask vector as $M_{\text{mask}} = (m_{\text{mask}_{t_1}}, m_{\text{mask}_{t_2}}, \ldots, m_{\text{mask}_{t_n}}) \in R^{n\times q}$, the element value of which gets from $\{0, 1\}^q$. M_{mask} is used to indicate whether the value of X has been observed, which is defined as

$$m_{\text{mask}_{t_i}} = \begin{cases} 1, & \text{if } x_{t_i}^j \text{ is observed} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

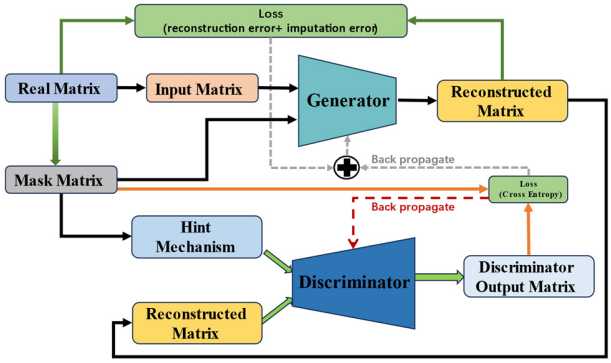


Fig. 1. Architecture of GAIN.

Furthermore, the definition for the missingness rate of the data is

$$\varphi = \left(\sum_{i=1}^{n}\sum_{j=1}^{q} 1 - M_{t_i}^j\right) \Big/ (n \times d). \tag{2}$$

B. Generative Adversarial Imputation Nets

Yoon et al. [13] introduced GAIN, a deep generative model comprising a generator (G), a discriminator (D), and a hinting mechanism (H). The GAIN architecture, depicted in Fig. 1, involves G capturing the data distribution and D assessing the probabilities of sample origin. Adversarial training, guided by H, leads to the convergence of G and D to a Nash equilibrium. This training alternation involves G attempting to deceive D by mapping a prior distribution to the data space, while the task of D is to differentiate the input source. Nash equilibrium is achieved when D cannot accurately determine the data source and G cannot generate further fake samples. Through this adversarial process, GAIN produces samples indistinguishable from the original data distribution.

The distinctive feature of GAIN, compared to conventional models, lies in its adoption of a min **i** max adversarial game. G takes a random variable (z) as input, generating synthetic samples $G(z)$ with the goal of minimizing the generation error. D takes a sample as input, providing the probability of that sample being real data, aiming to minimize its own error rate. The min-max objective of GAIN is defined as

$$\min_G \max_D V(D, G) = E_{x\sim P_{\text{data}}(x)} \log(D(x)) + E_{x\sim P_z(z)} \log(D(G(z))) \tag{3}$$

where x represents real data, $P_{\text{data}}(x)$ denotes the distribution of real data, and $P_z(z)$ represents the assumed prior data distribution.

III. Stacked Adaptive Mean Iteration Imputation Framework

This part introduces the stacked adaptive mean iteration imputation framework for reconstructing missing time series data in industrial processes. This framework utilizes mean

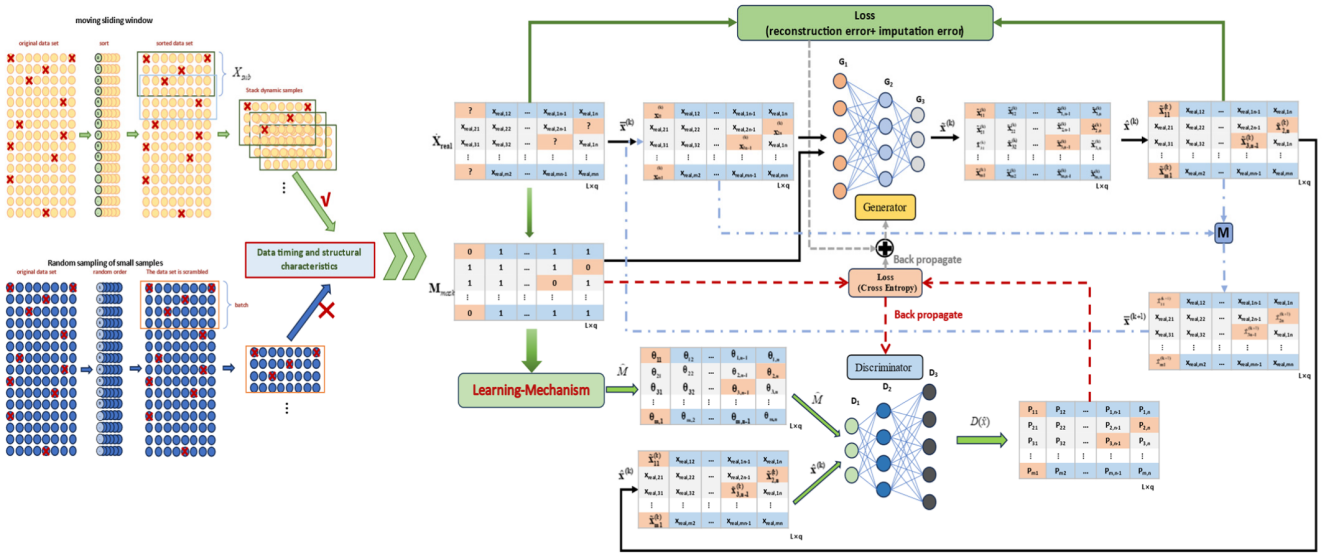


Fig. 2. Stacked adaptive mean iterative network architecture diagram. The left section employs a moving window, replacing the random sampling small sample method employed in the original GAIN and maintaining the temporal and structural features of the data. The detailed schematic on the right illustrates the network, with blue blocks labeled "*M*" indicating mean computation, and light green modules representing the adaptive learning mechanism module.

recursive updates of input values, ensuring the transmission of imputation information during training while preserving the overall trend of the data. In addition, the introduction of stacked dynamic samples increases the model training depth, enhancing its understanding of time series data. SAMIIF is an unsupervised deep generative model composed of a generator G, a discriminator D, and an adaptive learning mechanism, aimed at enhancing the imputation performance of time series data. Fig. 2 presents the overall process and implementation of the SAMIIF model, which encompasses the following three steps: First, a moving window is employed to obtain continuous subsequences as inputs to the model and retain temporal information of the data, which can enable the model to learn dependencies between sequences. Next, within each sub-block, a training approach involving dynamic sample overlay is employed to increase the model training depth, thereby enhancing its understanding of time series data. Finally, by using the output of one sub-block and the mean of its input as the input to the next sub-block, a mean recursive update is achieved, allowing imputation information to propagate during training and thus improving the efficiency and accuracy of the model inputs. The following parts are the detailed introduction to the various parts of the network structure.

A. Moving Sliding Window

The original GAIN employed a strategy of randomly sampling small batches during training, achieved by randomly sorting the original dataset and selecting batches from it. This approach effectively reduces computational costs and the risk of model overfitting. Particularly, it is beneficial when handling large-scale datasets. It saves significant time and resources.

However, when dealing with time series data, random sampling may disrupt the temporal sequence of samples, hindering the model's ability to learn temporal dependencies. To address this issue, we introduced the moving window method, as illustrated in Fig. 3, where time series data are partitioned into fixed-length windows used as training samples.

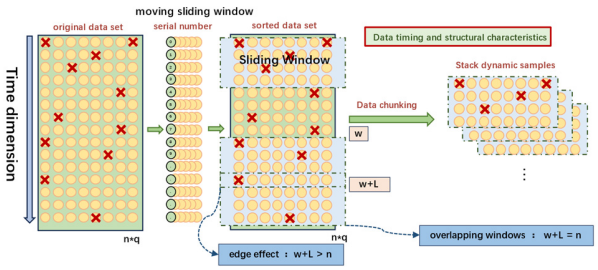


Fig. 3. Preserving data temporal and structural characteristics through the use of moving window technology, mitigating edge effects by employing overlapping windows.

These windows move forward with a fixed step size to maintain the temporal order of the data, effectively learning temporal dependencies. To mitigate edge effects, overlapping windows are adopted, ensuring that the model fully leverages complete window data during training, thereby improving performance. The data sequence in the sub-block can be represented as

$$\begin{aligned} X_{\text{sub}} &= \text{SlidingWindow}\left(X_{\text{real}}\right) \\ &= \left[\left(x_1, \ldots, x_L\right)^T, \ldots, \left(x_{n-L+1}, \ldots, x_n\right)^T\right]^T \end{aligned} \tag{4}$$

where $X_{\text{sub}} \in R^{L\times q}$ represents the sub-block dataset, $x \in R^{q\times 1}$ is an individual sample, q represents the total length of the time series data, L is the sliding window length, the starting position of the window is denoted as w, and the stride length is s. For the subsequent occurrences of x, unless otherwise specified, it refers to the sub-block data.

B. Generator

In SAMIIF, both G and D undergo adversarial training concurrently. G aims to capture the underlying data distribution, while D is used to estimate the probability that a sample comes from the underlying data rather than from the output of G. Both components employ a multilayer perceptron structure. Generator G takes $\bar{x}^{(k)}$, M_{mask}, and random noise z as inputs and the output is $\tilde{x}^{(k)}$.

This study employs a mean iteration method to update the generator input. Specifically, in the initial iteration process, random noise is used to initialize the places with missing values, resulting in the complete dataset $\bar{x}^{(1)}$. At the kth iteration, the mean of the output $\hat{x}^{(k)}$ of the sub-block and the input $\bar{x}^{(k)}$ is used as the input $\bar{x}^{(k+1)}$ for the next iteration. The specific process is illustrated in Fig. 2. Here, $\dot{X}_{\text{real}}$ represents the incomplete underlying dataset containing missing values, and M_{mask} represents the mask vector dependent on $\dot{X}_{\text{real}}$.

By utilizing the mean iteration method to update the generator input, we effectively leverage the mean to convey completion information, making the completion process more stable and reliable. Through mathematical equations, we can precisely define the iterative input update process depicted in Fig. 2

$$\begin{aligned} \bar{x}^{(1)} &= (1 - M_{\text{mask}}) \odot \dot{X}_{\text{real}} + M_{\text{mask}} \odot z \\ \bar{x}^{(k+1)} &= \bar{x}^{(k)}_{\text{median}} = \left(\bar{x}^{(k)} + \hat{x}^{(k)}\right)/2k = 1,2,\ldots,. \end{aligned} \tag{5}$$

Next, we provide the definition of the reconstruction values $\hat{x}^{(k)}$

$$\begin{aligned} \tilde{x}^{(1)} &= G\left(\bar{x}^{(1)}, M_{\text{mask}}, (1 - M_{\text{mask}}) \odot z\right) \\ \tilde{x}^{(k)} &= G\left(\bar{x}^{(k)}, M_{\text{mask}}, (1 - M_{\text{mask}}) \odot \bar{x}^{(k)}\right) \quad k = 1,2,\ldots, \\ \hat{x}^{(k)} &= M_{\text{mask}} \odot \dot{X}_{\text{real}} + (1 - M_{\text{mask}}) \odot \tilde{x}^{(k)} \quad k = 1,2,\ldots,. \end{aligned} \tag{6}$$

Here, $\odot$ represents element-wise multiplication, where $\bar{x}^{(k)}$ corresponds to the input vector of the generator, and $\tilde{x}^{(k)}$ corresponds to the reconstructed output sample values for each component generated by the generator. It is noteworthy that G outputs a value for each component, regardless of whether it is missing. $\hat{x}^{(k)}$ corresponds to the preserved real values of the underlying data, with the generator output values only replacing the missing values at their respective positions (resulting in a complete data vector after filling).

The mean recursive update demonstrates significant advantages in time series data processing. First, it preserves the temporal relationships of the original time series data. By computing the mean for each sub-block and integrating the model prediction results with the previous input, it aligns the completion values more closely with the trends in the time series data. Second, updating the input with the mean in each sub-block leads to smoother changes in completion values between adjacent sub-blocks, reducing drastic fluctuations in prediction results and resulting in a more continuous and natural overall prediction outcome. In addition, it helps mitigate the prediction result fluctuations caused by random noise or model errors, enhancing the stability and reliability of completion values. At last, calculating the mean can be seen as giving equal weight (50%) to both previous and current reconstructed values, expediting the training process. In summary, mean recursive updating effectively preserves the characteristics of time series data, resulting in more accurate and reliable completion values. Furthermore, the stacked dynamic sampling strategy is adopted to increase the model training depth, enabling it to more effectively comprehend and learn the complex characteristics of time series data. This is particularly crucial when complex temporal dependencies exist in the data, thereby enhancing the model's understanding of time series data.

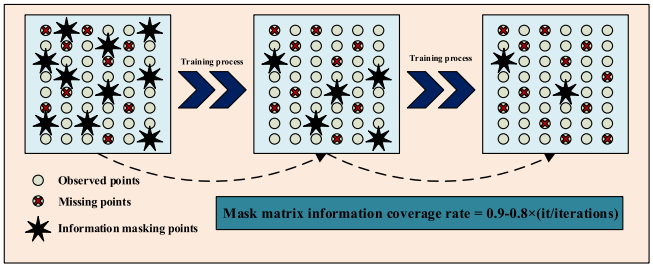


Fig. 4. Adaptive learning mechanism detailed structural schematic.

C. Adaptive-Learning Mechanism

To ensure that the data distribution learned by the generator aligns with expectations and to avoid an imbalance in the capabilities of the generator and discriminator, an adaptive learning mechanism is introduced. The structure of this mechanism is illustrated in Fig. 4, with its core concept revolving around gradually increasing the additional information provided to the discriminator during its training process. This augmentation aims to reduce the unknown masking points in the true mask vector that the discriminator encounters, thereby making its output more closely approximate the mask vector M_{mask}.

The adaptive learning mechanism involves a dynamic numerical value within the range of $\{0, 1\}^n$, dependent on the mask vector M_{mask} derived from the underlying data. Specifically, for each output $\tilde{x}^{(k)}$ from G, a sample θ is extracted based on the distribution of $\widehat{M}|M_{\text{mask}} = m_{\text{mask}}$, serving as an additional input to D. Each component $P_{i,j}$ of D output from$D(\hat{x})$ represents the probability of observing the ith component under the given conditions $\hat{X} = \hat{x}$ and $\widehat{M} = \theta$.

Defining a random variable vector $A = (\alpha_1, \ldots, \alpha_n) \in \{0, 1\}^n$, where α is a random value within the range $[0, 1)$, we concurrently introduce the mask information coverage ε, aiming to control the amount of information contained in $\widehat{M} = (\theta_1, \theta_2, \ldots, \theta_n) \in \{0, 0.5, 1\}^n$ by M_{mask}. The specific definition is given by the following formula:

$$\varepsilon = 0.9 - 0.8 \times (it/iterations) \tag{7}$$

$$\alpha_k = \begin{cases} 0, & \text{if } \alpha_k < \varepsilon \\ 1, & \text{if } \alpha_k > \varepsilon \end{cases} \tag{8}$$

$$\widehat{M} = A \odot M_{\text{mask}} + 0.5\,(1 - A)\,. \tag{9}$$

Then, we can derive

$$\theta_k = \begin{cases} m_{\text{mask}}, & \alpha_k = 1 \\ 0.5, & \alpha_k = 0. \end{cases} \tag{10}$$

When θ_k is equal to 0.5, the adaptive learning mechanism does not provide additional information to D. It is important to note that as the mask coverage ε increases, the amount of additional information given to D decreases, and ε gradually decreases as the training process progresses.

D. Discriminator

We define the discriminator D as a function: $D : \dot{\chi}_{\text{real}} \rightarrow [0, 1]^n$, where the ith component of $D(\hat{x})$ represents the probability assigned by the discriminator to the i thcomponent being a true value. The discriminator D is a binary classifier with the primary task of distinguishing whether the data originate from the underlying distribution or from the generator output. It takes the output $\hat{x}^{(k)}$ of the generator as input, which includes both the observed and the predicted values. The discriminator calculates the probability by assessing the authenticity of the components in $\hat{x}^{(k)}$, ultimately outputting a value between 0 and 1. Its objective is to approach a probability of 1 for real samples and a probability of 0 for generated samples. Through continuous training, the discriminator gradually optimizes its ability to more effectively distinguish between real and fake samples. This is analogous to predicting the mask vector M_{mask}. Ideally, we expect the discriminator output to closely align with M_{mask}. Therefore, in an ideal scenario, the discriminator output should be equivalent to M_{mask} that is determined by the original incomplete dataset.

E. Definition of Model Loss

For the model training, the generator and discriminator undergo adversarial training, enabling the generator to produce time series data that better conforms to the distribution of real data. The following provides detailed definitions of the loss functions for the generator and discriminator. We train D to optimize for assigning the accurate label to both training instances and generated samples from G. We concurrently train G to minimize the generator loss function. This establishes a minimax game. Therefore, our value function [13], [36] can be described as

$$\min_G \max_D V(D, G) = E_{x \sim P_{\text{data}}(x)} \log\left(D(x)\right) + E_{x \sim P_z(z)} \log\left(1 - D\left(G(z)\right)\right). \tag{11}$$

This configuration is very similar to the standard GAN. We define a binary cross-entropy $J_D(m_{\text{mask}}, \hat{m}, \varepsilon)$ as

$$J_D\left(m_{\text{mask}}, \hat{m}, \varepsilon\right) = \sum_{i=0} \left[m_{\text{mask}_i} \log\left(\hat{m}_i\right) + \left(1 - m_{\text{mask}_i}\right) \log\left(\hat{m}_i\right)\right]. \tag{12}$$

D is then trained according to

$$\min_D - \sum_{j=1}^{k_D} J_D\left(m_{\text{mask}}(j), \hat{m}(j), \varepsilon\right) \tag{13}$$

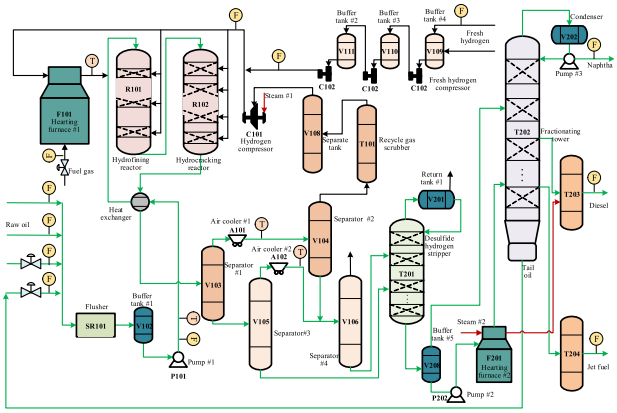


Fig. 5. Schematic of the hydrocracking process.

where k_D represents the number of samples in a mini-batch, recalling that $\hat{m}(j) = D(\hat{x}(j), \widehat{m}(j))$. The loss function of the generator consists of two distinct components: the completion loss $J_c(m_{\text{mask}}, \hat{m}, \varepsilon)$ at missing values ($m_{\text{mask}_i} = 0$) and the reconstruction loss $J_R(\bar{x}, \hat{x})$ at observed values ($m_{\text{mask}_i} = 1$), which are defined as

$$J_c\left(m_{\text{mask}}, \hat{m}, \varepsilon\right) = -\sum_{i=0} \left(1 - m_{\text{mask}_i}\right) \log\left(\hat{m}_i\right) \Bigg] \tag{14}$$

$$J_R\left(\bar{x}, \hat{x}\right) = \sum_{i=1}^{d} m_{\text{mask}} J_r\left(\bar{x}_i, \hat{x}_i\right) \tag{15}$$

where

$$J_r\left(\bar{x}_i, \hat{x}_i\right) = \begin{cases} -\bar{x}_i \log\left(\hat{x}_i\right), & \text{if } \bar{x}_i \text{ is binary} \\ \left(\hat{x}_i - \bar{x}_i\right)^2, & \text{if } \bar{x}_i \text{ is continuous.} \end{cases} \tag{16}$$

G is then trained with mini-batches of size k_G according to

$$\min_G \sum_{j=1}^{k_G} J_c\left(m_{\text{mask}}, \hat{m}, \varepsilon\right) + \tau J_R\left(\bar{x}, \hat{x}\right) \tag{17}$$

where τ is a hyper-parameter.

The pseudocode for SAMIIF is represented in Algorithm 1. The complete dataset will be obtained after SAMIIF training, and the final imputation results can be expressed as

$$X_{\text{imput}} = \hat{x}^{(\text{epoch})}. \tag{18}$$

IV. Industrial Applications

In this section, experimental validation of the proposed methodology was conducted on two real-world industrial datasets. A comprehensive analysis of the experimental results was performed, and comparisons were made with other benchmark models.

A. Datasets

1) Hydrocracking Process: Hydrocracking is a vital refining technique used to convert heavy petroleum fractions into lighter, high-quality products like gasoline. The process is depicted in Fig. 5. In challenging production environments with high temperatures and pressures, unstable sensor communication and equipment failures may lead to missing

Algorithm 1 Recover Time-Series Missing Data Using SAMIIF

Data: Real data with missing values X_{real}, mask position matrix M_{mask}, sliding window length L, number of samples n, Adaptive Learning Matrix A
Output: Imputed data set $\hat{x}^{(epoch)}$

Data chunkling: $X_{sub} = SlidingWindow\ (X_{real})$
while training loss has not converged **do**
  (1) Discriminator optimization
  Draw Δt samples from the dateset $\{(\hat{x}(j), \theta(j))\}_{j=1}^{n-L+1}$
  for $k \longleftarrow 1$ **to** *epoch* **do**
    $\hat{x}^{(k)} \leftarrow G\left(\bar{x}^{(k)}, M_{mask}\right)$
    $\hat{x}^{(k)} = M_{mask} \odot x_{real} + (1 - M_{mask}) \odot \dot{x}^{(k)}$
    $A \leftarrow \varepsilon = 0.9 - 0.8 \times (k/epoch)$
    $\hat{M} = A \odot M_{mask} + 0.5(1 - A)$
  end
  Update D using stochastic gradient descent (SGD)
  $\nabla_D - \sum_{j=1}^{T-\Delta t+1} J_D\left(m_{mask}(j), \tilde{m}(j), \varepsilon\right)$
  (2) Generator optimization
  Draw Δt samples from the dataset $\{(\hat{x}(j), \theta(j))\}_{j=1}^{n-L+1}$
  for $k \longleftarrow 1$ **to** *epoch* **do**
    $\bar{x}^{(1)} = (1 - M_{mask}) \odot \dot{X}_{real} + M_{mask} \odot z$
    $\bar{x}^{(k+1)} = \bar{x}_{median}^{(k)} = \left(\bar{x}^{(k)} + \bar{x}^{(k)}\right)/2$
    $A \leftarrow \varepsilon = 0.9 - 0.8 \times (k/epoch)$
    $\bar{M} = A \odot M_{mask} + 0.5(1 - A)$
  end
  Update G using stochastic gradient descent (SGD)
  $\nabla_G \sum_{j=1}^{n-L+1} J_c\left(m_{mask}, \hat{m}, \varepsilon\right) + \tau J_R(\bar{x}, \hat{x})$
end

time series data. Consequently, a crucial task involves predicting and restoring missing data to ensure a complete dataset based on the available information in adverse conditions [45]. The data are collected from a petrochemical plant in China. There are a total of 43 measured parameters. The sampling frequency is about a sample every 2 h from December 2015 to November 2018 with a total of 2606 data samples.

2) Sulfur Recovery Unit (SRU): SRU is commonly used in petroleum refining and natural gas processing to extract sulfur from sulfur-containing gases. Their primary purpose is to reduce sulfur emissions, prevent environmental pollution, and recover sulfur as a by-product. Under harsh environmental conditions, sensor failures or communication issues may lead to data loss. This dataset consists of 10 085 data samples collected by seven SRU sensors. A detailed description of this dataset can be found in [46].

B. Baseline Methods

This study selected eight models for comparative analysis from statistical, machine learning, and neural network-based approaches to demonstrate the superiority of the SAMIIF model in data imputation. The models were implemented using the Python packages *TensorFlow* and *FancyImpute*.[1] The experimental setup for the neural network-based model was configured according to the corresponding literature for each method.

TABLE I
PARAMETER CONFIGURATION DETAILS FOR DIFFERENT METHODS IN THE HYDROCRACKING PROCESS

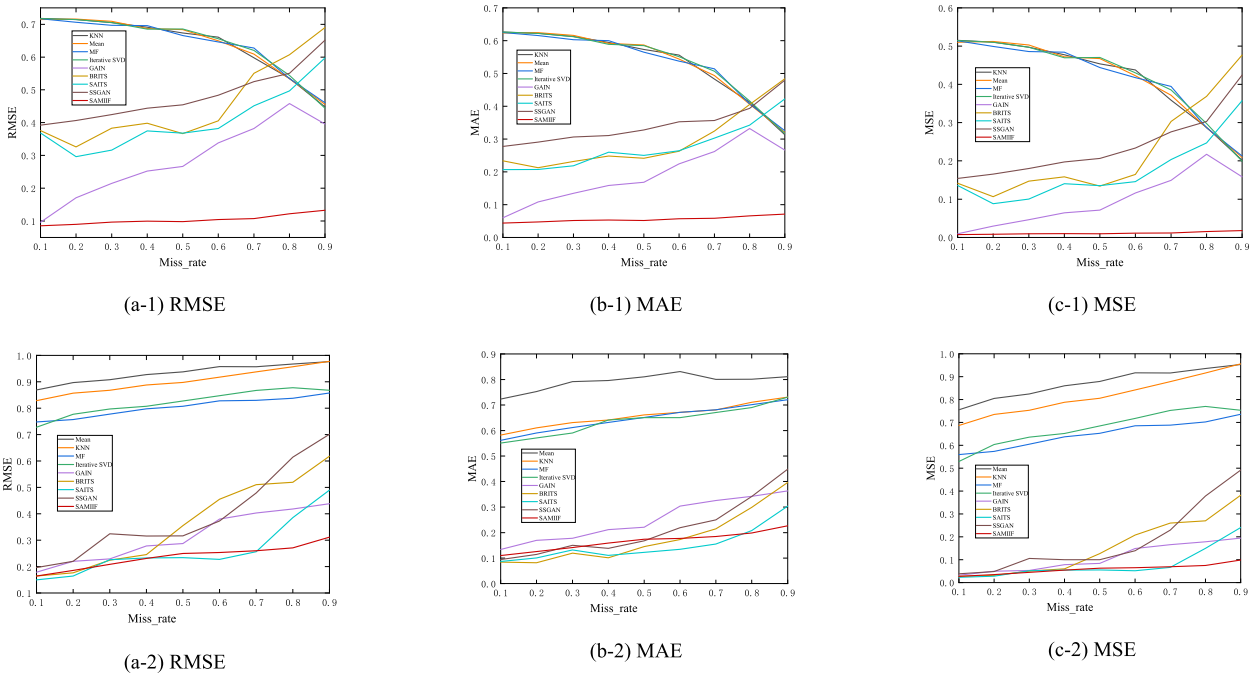
Methods	Parameters
KNN	{k=20}
MF	{rank=40,lr=0.01,max_iters=50}
Iterative SVD	{rank=10,convergence_threshold=0.00001, max_iters=200}
BRITS	{lr=0.001,epochs=90,patience=10,batch_size=32, weight_decay=0.00001
SAITS	{n_layers=2,d_model=256,d_inner=128, n_head=4,dropout=0.1,epochs=70}
GAIN	{batch_size=64,alpha=0.1,iterations=10000}
SSGAN	{n_steps=30,rnn_hidden_size=30,lr=0.0001, weight_decay=0.00001,batch_size=32,epochs=100}
SAMIIF	{batch_size=128,alpha=0.1,iterations=5000, stride=60,n=5}

1) *Mean:* Fills missing values with the global average of the variable.
2) *KNN [26]:* Imputes missing values by averaging the nearest neighboring samples. It identifies the K closest samples to the missing value and estimates the missing value based on the values of these K neighbors.
3) *MF [25]:* Decomposes an incomplete matrix into two low-rank matrices and then uses feature inference through methods like gradient descent to estimate missing values.
4) *Iterative SVD [47]:* A MF method used for estimating missing data by iteratively approximating the missing values in a matrix.
5) *Bidirectional Recurrent Imputation for Time Series (BRITS) [35]:* A time series imputation method based on recursive neural networks, directly learning missing values within a bidirectional recurrent dynamic system without requiring specific assumptions.
6) *Self-Attention-based Imputation for Time Series (SAITS) [37]:* A method based on self-attention mechanisms. Through joint optimization training, it utilizes self-attention blocks with two diagonal masks to learn missing values in multivariate time series, explicitly capturing time dependencies and feature correlations.
7) *GAIN [13]:* Using GAN to impute missing values, along with the incorporation of a hint mechanism.
8) *SSGAN Model [43]:* A **SSGAN** that enhances imputation accuracy by leveraging label information and a temporal reminder matrix.

[1] https://github.com/iskandr/fancyimpute

TABLE II
AVERAGE PERFORMANCE OF NINE METHODS IN HYDROCRACKING AND SRU SCENARIOS

Datasets		Hydrocracking								SRU							
Method	Miss_rate	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Mean	RMSE	0.7144	0.7158	0.7091	0.6870	0.6836	0.6499	0.6092	0.5340	0.8690	0.8974	0.9081	0.9276	0.9377	0.9578	0.9573	0.9676
	MAE	0.6233	0.6245	0.6164	0.5918	0.5871	0.5441	0.4935	0.4063	0.7232	0.7531	0.7917	0.7959	0.8106	0.8313	0.8007	0.8009
	MSE	0.5104	0.5124	0.5030	0.4721	0.4675	0.4228	0.3717	0.2867	0.7552	0.8053	0.8247	0.8605	0.8793	0.9175	0.9165	0.9362
KNN	RMSE	0.7177	0.7145	0.7050	0.6909	0.6734	0.6611	0.5980	0.5353	0.8289	0.8575	0.8680	0.8879	0.8977	0.9177	0.9376	0.9574
	MAE	0.6264	0.6226	0.6126	0.5946	0.5735	0.5564	0.4830	0.4125	0.7232	0.753	0.7917	0.7959	0.8106	0.8313	0.8007	0.8009
	MSE	0.5151	0.5106	0.4972	0.4775	0.4537	0.4378	0.3589	0.2874	0.7552	0.8053	0.8247	0.8605	0.8793	0.9175	0.9165	0.9362
MF	RMSE	0.7167	0.7064	0.6970	0.6957	0.6659	0.6461	0.6277	0.5343	0.7480	0.7574	0.7780	0.7980	0.8079	0.8280	0.8298	0.8379
	MAE	0.6248	0.6156	0.6030	0.6002	0.5650	0.5377	0.5143	0.4075	0.5618	0.5906	0.6116	0.6315	0.6512	0.6715	0.6812	0.7014
	MSE	0.5137	0.4991	0.4861	0.4842	0.4440	0.4178	0.3949	0.2866	0.5595	0.5736	0.6053	0.6368	0.6526	0.6856	0.6885	0.7021
Iterative SVD	RMSE	0.7175	0.7144	0.7049	0.6854	0.6859	0.6559	0.6204	0.5442	0.7278	0.7771	0.7972	0.8076	0.8274	0.8471	0.8674	0.8776
	MAE	0.6266	0.6222	0.6121	0.5888	0.5851	0.5515	0.5070	0.4159	0.5512	0.5707	0.5903	0.6409	0.6507	0.6505	0.6707	0.6905
	MSE	0.5149	0.5103	0.4971	0.4699	0.4706	0.4304	0.3854	0.2976	0.5297	0.6039	0.6355	0.6523	0.6845	0.7177	0.7524	0.7703
BRITS	RMSE	0.3759	0.3262	0.3833	0.3981	0.3664	0.4055	0.5505	0.6070	0.1640	0.1764	0.2246	0.2457	0.3555	0.4555	0.5103	0.5197
	MAE	0.2336	0.2123	0.2317	0.2483	0.2414	0.2628	0.3244	0.4054	0.0841	0.0821	0.1192	0.1013	0.1453	0.1725	0.2147	0.2986
	MSE	0.1415	0.1064	0.1469	0.1585	0.1343	0.1645	0.3031	0.3685	0.0269	0.0311	0.0504	0.0604	0.1264	0.2075	0.2604	0.2701
SAITS	RMSE	0.3684	0.2965	0.3164	0.3748	0.3680	0.3819	0.4513	0.4970	0.1493	0.1640	0.2264	0.2330	0.2336	0.2274	0.2566	0.3858
	MAE	0.2068	0.2073	0.2185	0.2596	0.2504	0.2644	0.3029	0.3424	0.0872	0.1004	0.1319	0.1104	0.1226	0.1344	0.1553	0.2082
	MSE	0.1358	0.0879	0.1002	0.1406	0.1355	0.1460	0.2038	0.2472	0.0225	0.0277	0.0515	0.0545	0.0548	0.0518	0.0660	0.1495
GAIN	RMSE	0.0964	0.1707	0.2148	0.2524	0.2663	0.3379	0.3821	0.4582	0.1787	0.2197	0.2291	0.2782	0.2875	0.3800	0.4029	0.4180
	MAE	0.0595	0.1079	0.1344	0.1583	0.1679	0.2242	0.2621	0.3322	0.1338	0.1694	0.1775	0.2119	0.2208	0.3037	0.3254	0.3424
	MSE	0.0094	0.0293	0.0463	0.0644	0.0714	0.1162	0.1493	0.2176	0.0323	0.0497	0.0530	0.0780	0.0836	0.1490	0.1659	0.1783
SSGAN	RMSE	0.3924	0.4066	0.4245	0.4442	0.4546	0.4835	0.5250	0.5505	0.1963	0.2201	0.3244	0.3160	0.3162	0.3730	0.4787	0.6150
	MAE	0.2775	0.2908	0.3062	0.3108	0.3279	0.3525	0.3565	0.3936	0.0944	0.1147	0.1502	0.1380	0.1677	0.2190	0.2500	0.3416
	MSE	0.1540	0.1654	0.1802	0.1973	0.2066	0.2338	0.2757	0.3031	0.0386	0.0484	0.1053	0.0999	0.1000	0.1392	0.2292	0.3784
SAMIIF	RMSE	**0.0854**	**0.0900**	**0.0965**	**0.0993**	**0.0979**	**0.1044**	**0.1072**	**0.1221**	**0.1644**	**0.1852**	**0.2084**	**0.2305**	**0.2498**	**0.2532**	**0.2603**	**0.2712**
	MAE	**0.0438**	**0.0472**	**0.0514**	**0.0531**	**0.0513**	**0.0566**	**0.0582**	**0.0656**	**0.1100**	**0.1258**	**0.1410**	**0.1590**	**0.1737**	**0.1770**	**0.1848**	**0.1982**
	MSE	**0.0074**	**0.0081**	**0.0093**	**0.0099**	**0.0096**	**0.0110**	**0.0115**	**0.0151**	**0.0273**	**0.0349**	**0.0444**	**0.0543**	**0.0632**	**0.0647**	**0.0695**	**0.0749**



(a-1) RMSE (b-1) MAE (c-1) MSE

(a-2) RMSE (b-2) MAE (c-2) MSE

Fig. 6. RMSE, MAE, and mse results of nine methods under different missing rates in the hydrocracking and SRU processes, where (*-1) indicates the results for the hydrocracking dataset and (*-2) indicates the results for the SRU dataset. Here, * represents (a), (b), and (c). (a-1) RMSE, (a-2) RMSE, (b-1) MAE, (b-2) MAE, (c-1) mse, and (c-2) mse.

C. Experimental Results and Analysis

We compared our model with eight other imputation algorithms on the two datasets to verify its effectiveness. Testing data points were randomly removed at rates ranging from 10% to 90% to assess imputation performance under different missing rates. The missing rate represents the proportion of missing data points relative to the total dataset, with a higher rate indicating a greater imputation challenge. Prior to experimentation, we normalized the dataset. To ensure accuracy, we conducted ten repetitions of the experiments and averaged the results. Three metrics (RMSE, MAE, and mse) [1], [37] were used to evaluate performance. The optimal parameters obtained through trial and error are presented in Table I. Experimental results for different missing rates, using the obtained parameter set, are summarized in Table II. Fig. 6 presents detailed results for the nine methods.

As can be seen from the results of Table II and Fig. 6, it can be concluded that statistical and machine learning methods, such as KNN, mean, MF, and iterative SVD, exhibit no significant differences in performance. The KNN algorithm utilizes information from neighboring samples, while mean imputation employs a simple mean-based imputation method. Both methods perform similarly on the hydrocracking dataset but are less effective on the SRU dataset. MF and iterative

SVD are matrix imputation methods. The former directly decomposes the original matrix, while the latter approximates singular value decomposition through iterative steps. Both methods are relatively well-suited to the distribution characteristics of the datasets, resulting in similar outcomes across both datasets.

In contrast, the BRITS algorithm demonstrates superior capability in handling complex data distributions and correlations, especially at lower missing rates. However, its performance significantly deteriorates at higher missing rates due to its nature as a supervised learning algorithm, which struggles to learn the original data distribution in datasets with many missing values. In comparison, the SAITS algorithm performs better than BRITS at high missing rates but shows decreased performance at extremely high missing rates (0.9).

The GAIN and SSGAN algorithms excel at capturing complex data distributions and structures, particularly at low missing rates. Despite facing greater challenges with increasing missing rates, GAIN maintains high accuracy even at a missing rate of 0.9, possibly due to its ability to better fit data patterns and features in specific scenarios. In contrast, SSGAN performance declines noticeably at higher missing rates, potentially due to overfitting on the training set caused by the high missing rate.

In comparison, the proposed SAMIIF algorithm outperforms the other eight algorithms in the experiments. By adaptively prompting the discriminator, SAMIIF effectively enhances the discriminator's capability, compelling the generator to produce data closer to the real samples. In addition, the adaptive mean iteration updates input values, resulting in smoother model outputs and increased stability. SAMIIF demonstrates outstanding imputation accuracy even at higher missing rates.

V. Conclusion

To address the issue of missing temporal data in industrial processes, we propose a stacked adaptive median iterative imputation network based on GANs. First, we partition the ordered dataset using a moving window approach. Subsequently, we employ a GAN to learn the data distribution and predict missing values. Concurrently, a mean iteration is utilized to update inputs, facilitating effective information transfer during the training process to enhance the model's efficiency and accuracy. In addition, we introduce an adaptive learning mechanism framework to adaptively provide learning information to the discriminator during training, preventing imbalance between the generator and discriminator in adversarial games. This approach aims to avoid issues such as model degradation, pattern collapse, and convergence problems. In several real industrial process datasets, we simulated incomplete datasets with different missing rates and compared our method with others, verifying its superiority and stability. This study primarily focuses on addressing the issue of MCAR-type missing data in industrial time series. In the future, we will further investigate methods for handling MNAR and MAR types of missing data, expanding the applicability of the existing model. In addition, we will explore data imputation strategies in scenarios where multiple missing mechanisms coexist, thereby enhancing the robustness of the model.

References

[1] X. Yuan et al., "Hierarchical self-attention network for industrial data series modeling with different sampling rates between the input and output sequences," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 2024.

[2] Y. Lyu, L. Zhou, Y. Cong, H. Zheng, and Z. Song, "Multirate mixture probability principal component analysis for process monitoring in multimode processes," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 2, pp. 2027–2038, Apr. 2024.

[3] N. U. Okafor and D. T. Delaney, "Missing data imputation on IoT sensor networks: Implications for on-site sensor calibration," *IEEE Sensors J.*, vol. 21, no. 20, pp. 22833–22845, Oct. 2021.

[4] X. Yuan et al., "A cloud-edge collaborative framework for adaptive quality prediction modeling in IIoT," *IEEE Sensors J.*, early access, Aug. 2024.

[5] Q. Ni and X. Cao, "MBGAN: An improved generative adversarial network with multi-head self-attention and bidirectional RNN for time series imputation," *Eng. Appl. Artif. Intell.*, vol. 115, Oct. 2022, Art. no. 105232.

[6] Z. Pan, Y. Wang, K. Wang, H. Chen, C. Yang, and W. Gui, "Imputation of missing values in time series using an adaptive-learned median-filled deep autoencoder," *IEEE Trans. Cybern.*, vol. 53, no. 2, pp. 695–706, Feb. 2023.

[7] R. Wu, S. D. Hamshaw, L. Yang, D. W. Kincaid, R. Etheridge, and A. Ghasemkhani, "Data imputation for multivariate time series sensor data with large gaps of missing data," *IEEE Sensors J.*, vol. 22, no. 11, pp. 10671–10683, Jun. 2022.

[8] X. Yuan et al., "Variable correlation analysis-based convolutional neural network for far topological feature extraction and industrial predictive modeling," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–10, 2024.

[9] X. Yuan et al., "Quality prediction modeling for industrial processes using multiscale attention-based convolutional neural network," *IEEE Trans. Cybern.*, vol. 54, no. 5, pp. 2696–2707, May 2024.

[10] X. Yuan, W. Xu, Y. Wang, C. Yang, and W. Gui, "A deep residual PLS for data-driven quality prediction modeling in industrial process," *IEEE/CAA J. Autom. Sinica*, vol. 11, no. 8, pp. 1777–1785, Aug. 2024.

[11] J. Du, M. Hu, and W. Zhang, "Missing data problem in the monitoring system: A review," *IEEE Sensors J.*, vol. 20, no. 23, pp. 13984–13998, Dec. 2020.

[12] Z.-W. Zhang, H.-P. Tian, L.-Z. Yan, A. Martin, and K. Zhou, "Learning a credal classifier with optimized and adaptive multiestimation for missing data imputation," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 7, pp. 4092–4104, Jul. 2022.

[13] J. Yoon, J. Jordon, and M. Schaar, "GAIN: Missing data imputation using generative adversarial nets," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5689–5698.

[14] M. Jamshidian, S. Jalal, and C. Jansen, "MissMech: AnRPackage for testing homoscedasticity, multivariate normality, and missing completely at random (MCAR)," *J. Stat. Softw.*, vol. 56, no. 6, pp. 1–31, 2014.

[15] W.-C. Lin and C.-F. Tsai, "Missing value imputation: A review and analysis of the literature (2006–2017)," *Artif. Intell. Rev.*, vol. 53, no. 2, pp. 1487–1509, Feb. 2020.

[16] C. Ou et al., "Missing-data imputation with position-encoding denoising autoencoders for industrial processes," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–11, 2024.

[17] S. Yang, M. Dong, Y. Wang, and C. Xu, "Adversarial recurrent time series imputation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 4, pp. 1639–1650, Apr. 2023.

[18] A. Farhangfar, L. A. Kurgan, and W. Pedrycz, "A novel framework for imputation of missing values in databases," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 37, no. 5, pp. 692–709, Sep. 2007.

[19] K. Sarda, A. Yerudkar, and C. D. Vecchio, "Missing data imputation for real time-series data in a steel industry using generative adversarial networks," in *Proc. 47th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2021, pp. 1–6.

[20] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern classification with missing data: A review," *Neural Comput. Appl.*, vol. 19, no. 2, pp. 263–282, Mar. 2010.

[21] R. J. Little and D. B. Rubin, *Statistical Analysis With Missing Data*. Hoboken, NJ, USA: Wiley, 2019.

[22] R. A. Hughes, J. Heron, J. A. C. Sterne, and K. Tilling, "Accounting for missing data in statistical analyses: Multiple imputation is not always the answer," *Int. J. Epidemiol.*, vol. 48, no. 4, pp. 1294–1304, Aug. 2019.
[23] G. E. Batista and M. C. Monard, "A study of K-nearest neighbour as an imputation method," *HIS*, vol. 87, nos. 251–260, p. 48, Dec. 2002.
[24] S.-B. Roh, T.-C. Ahn, and W. Pedrycz, "The refinement of models with the aid of the fuzzy k-nearest neighbors approach," *IEEE Trans. Instrum. Meas.*, vol. 59, no. 3, pp. 604–615, Mar. 2010.
[25] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[26] O. Troyanskaya et al., "Missing value estimation methods for DNA microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, Jun. 2001.
[27] T. O. Ayodele, "Types of machine learning algorithms," in *New Advances in Machine Learning*, vol. 3, nos. 19–48. 2010, p. 5-1.
[28] M. M. Rahman and D. N. Davis, "Machine learning-based missing value imputation method for clinical datasets," in *Proc. IAENG Trans. Eng. Technol., Special Volume World Congr. Eng.*, 2012, pp. 245–257.
[29] F. Zhao et al., "Multiple imputation method of missing credit risk assessment data based on generative adversarial networks," *Appl. Soft Comput.*, vol. 126, Sep. 2022, Art. no. 109273.
[30] X. Yuan et al., "Attention-based interval aided networks for data modeling of heterogeneous sampling sequences with missing values in process industry," *IEEE Trans. Ind. Informat.*, vol. 20, no. 4, pp. 5253–5262, Apr. 2024.
[31] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu, "Time series classification with multivariate convolutional neural network," *IEEE Trans. Ind. Electron.*, vol. 66, no. 6, pp. 4788–4797, Jun. 2019.
[32] W. Li, L. Xiao, and B. Liao, "A finite-time convergent and noise-rejection recurrent neural network and its discretization for dynamic nonlinear equations solving," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3195–3207, Jul. 2020.
[33] B. Liao, Y. Zhang, and L. Jin, "Taylor O (h^3) discretization of ZNN models for dynamic equality-constrained quadratic programming with application to manipulators," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 225–237, Feb. 2016.
[34] Y. Zhou, J. Jiang, S.-H. Yang, L. He, and Y. Ding, "MuSDRI: Multi-seasonal decomposition based recurrent imputation for time series," *IEEE Sensors J.*, vol. 21, no. 20, pp. 23213–23223, Oct. 2021.
[35] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "BRITS: Bidirectional recurrent imputation for time series," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.
[36] W. Zhang, P. Zhang, Y. Yu, X. Li, S. A. Biancardo, and J. Zhang, "Missing data repairs for traffic flow with self-attention generative adversarial imputation net," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 7919–7930, May 2021.
[37] W. Du, D. Coté, and Y. Liu, "SAITS: Self-attention-based imputation for time series," *Exp. Syst. Appl.*, vol. 219, Jun. 2023, Art. no. 119619.
[38] W. Shao, Z. Ge, and Z. Song, "Semisupervised Bayesian Gaussian mixture models for non-Gaussian soft sensor," *IEEE Trans. Cybern.*, vol. 51, no. 7, pp. 3455–3468, Jul. 2021.
[39] W. Shao, Y. Li, W. Han, and D. Zhao, "Block-wise parallel semisupervised linear dynamical system for massive and inconsecutive time-series data with application to soft sensing," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–14, 2022.
[40] C. Xiao, W. Han, W. Shao, and D. Zhao, "Distributed semisupervised HMM for dynamic inferential sensor development," *IEEE Sensors J.*, vol. 23, no. 3, pp. 2737–2749, Feb. 2023.
[41] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–9.
[42] S. E. Awan, M. Bennamoun, F. Sohel, F. Sanfilippo, and G. Dwivedi, "Imputation of missing data with class imbalance using conditional generative adversarial networks," *Neurocomputing*, vol. 453, pp. 164–171, Sep. 2021.
[43] X. Miao, Y. Wu, J. Wang, Y. Gao, X. Mao, and J. Yin, "Generative semi-supervised learning for multivariate time series imputation," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 8983–8991.
[44] Y. Luo, X. Cai, Y. Zhang, and J. Xu, "Multivariate time series imputation with generative adversarial networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–12.
[45] X.-F. Yuan, L. Li, Y. Shardt, Y.-L. Wang, and C.-H. Yang, "Deep learning with spatiotemporal attention-based LSTM for industrial soft sensor model development," *IEEE Trans. Ind. Electron.*, vol. 68, no. 5, pp. 4404–4414, May 2020.
[46] L. Fortuna, S. Graziani, A. Rizzo, and M. G. Xibilia, *Soft Sensors for Monitoring and Control of Industrial Processes*. Berlin, Germany: Springer, 2007.
[47] Y. Liu and S. D. Brown, "Comparison of five iterative imputation methods for multivariate classification," *Chemometric Intell. Lab. Syst.*, vol. 120, pp. 106–115, Jan. 2013.

Xiaofeng Yuan (Senior Member, IEEE) received the B.Eng. degree in automation and the Ph.D. degree in control science and engineering from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2011 and 2016, respectively.

From 2014 to 2015, he was a Visiting Scholar with the Department of Chemical and Materials Engineering, University of Alberta, Edmonton, AB, Canada. He is currently a Professor at the School of Automation, Central South University, Changsha, China. His research interests include deep learning and artificial intelligence, machine learning and pattern recognition, industrial process soft sensor modeling, and process data analysis.

Dr. Yuan is an Associate Editor of IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT and IEEE SENSORS JOURNAL.

Jiale Zhang received the B.S. degree in automation from Yanshan University, Qinhuangdao, China, in 2022. She is currently pursuing the M.S. degree in electronic information with Central South University, Changsha, China.

Her research interests include deep learning, machine learning, soft sensor modeling, and process data mining.

Kai Wang (Member, IEEE) received the B.Eng. degree in automation and the Ph.D. degree in control science and engineering from the College of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2014 and 2019, respectively.

He was a Visiting Scholar with the Department of Chemical and Biological Engineering, University of British Columbia, Vancouver, BC, Canada. He is currently an Associate Professor at the School of Automation, Central South University, Changsha, China. He specializes in industrial data analytics, process health management, and machine learning.

Yalin Wang (Senior Member, IEEE) received the B.Eng. degree in automation and the Ph.D. degree in control science and engineering from the Department of Control Science and Engineering, Central South University, Changsha, China, in 1995 and 2001, respectively.

She is currently a Professor at the School of Automation, Central South University. Her research interests include modeling, optimization, and control for complex industrial processes, intelligent control, and process simulation.

Chunhua Yang (Fellow, IEEE) received the M.Eng. degree in automatic control engineering and the Ph.D. degree in control science and engineering from the Central South University, Changsha, China, in 1988 and 2002, respectively.

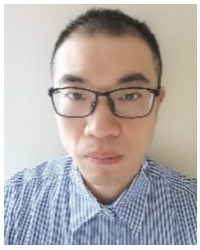
She was with the Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium, from 1999 to 2001. She is currently a Full Professor at Central South University. Her current research interests include modeling and optimal control of complex industrial processes, intelligent control systems, and fault-tolerant computing of real-time systems.

Weihua Gui (Member, IEEE) received the B.Eng. degree in electrical engineering and the M.S. degree in automatic control engineering from Central South University, Changsha, China, in 1976 and 1981, respectively.

Since 2013, he has been an Academician at the Chinese Academy of Engineering, Beijing, China. He is currently with the School of Automation, Central South University. His current research interests include modeling and optimal control of complex industrial processes, fault diagnoses, and distributed robust control.

Feifan Shen (Member, IEEE) received the B.Eng. degree in electronic information technology and instruments and the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2011 and 2016, respectively.

Since 2017, he has been with the School of Information Science and Engineering, NingboTech University, Ningbo, China, where he is currently an Associate Professor. His research interests include industrial big data, data-driven modeling, batch process control, process monitoring, soft sensor, and their applications in process industries.



Lingjian Ye (Member, IEEE) received the B.Eng. degree in chemical engineering and the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively.

From 2011 to 2020, he was with the School of Information Science and Engineering, Ningbo Institute of Technology, Zhejiang University. Since 2021, he has been a Professor at the School of Engineering, Huzhou University, Huzhou, China. His research interests include data-based process monitoring, plant-wide control, real-time optimization, and their applications in process systems.