Parallel Generative Adversarial Imputation Network for Multivariate Missing Time-Series Reconstruction and Its Application to Aeroengines

Song Ma, Zeng-Song Xu, and Tao Sun

***Abstract*— The reconstruction and imputation of missing values in multivariate time series (MTS) are pressing issues in the field of industrial artificial intelligence. To address this problem, an end-to-end deep learning model called the "time-series generative adversarial imputation network (TSGAIN) with preimputation, parallel convolution, and transformer encoder (PPCTE)" is proposed in this article. Specifically, the proposed TSGAIN framework is an improved generative adversarial network (GAN) for imputation of missing time series, which can consider both distribution and time information. Next, a preimputation module is advanced to obtain more accurate input information for the proposed model. Then, as two crucial components of the proposed model, the generator and discriminator are further improved. In particular, in terms of feature correlation, a parallel convolution (PC) module is designed, which further enhances the potential of the proposed model to extract feature correlation in multivariate data. In terms of temporality, by introducing the transformer encoder (TE) module with multihead attention mechanism, the ability of the proposed model to mine the temporal correlation of time-series data is further strengthened. Finally, a series of simulation experiments are carried out on the commercial modular aviation propulsion system simulation (C-MAPSS) experimental data provided by National Aeronautics and Space Administration (NASA), verifying the effectiveness of the proposed modules and the superiority of the proposed method in the imputation process of aeroengine missing sensor data.**

***Index Terms*— Data imputation and reconstruction, deep learning, industrial artificial intelligence, missing time series.**

I. Introduction

IN THE field of industrial artificial intelligence, multivariate time series (MTS) generated by multisensor measurements are important resources for prognostics and health management (PHM). For instance, aeroengine fault data can be used for fault diagnosis and control [1], [2], and aeroengine operating state data can be utilized to estimate the remaining useful life (RUL) [3], [4], [5]. In general, the first step of PHM techniques is to obtain the working state through advanced sensor techniques. Then, various diagnosis and prediction methods are developed to evaluate the health status of the machine for further operation planning and maintenance decisions [6]. With the development of sensor techniques, more sensors are applied for condition monitoring to meet the massive data requirements of PHM works [7]. However, sensors are vulnerable to harsh environments or electromagnetic interference, and data missing may occur in the measurement process, which will limit the subsequent health management work [8]. It is worth noting that high-quality MTS data can accurately reflect the actual health status and provide reliable information for PHM techniques, ultimately ensuring the safe operation of machinery. Therefore, it is a very important challenge to effectively deal with missing values in MTS data.

So far, the processing of missing data can be generally divided into two approaches: deletion method and imputation method. The deletion method discards missing data information to obtain a complete dataset, but it may reduce the amount of data and even lose key information. On the contrary, imputation methods usually estimate missing values through statistical analysis, similarity analysis, or model construction [9], which preserves all data. In general, the imputation method is applicable to most data loss processing processes, and the higher the accuracy of the imputation results, the smaller the impact on downstream tasks. Therefore, the imputation accuracy of imputation methods is critical. Nowadays, there are many common imputation methods. For example, one of the most original imputation methods is to replace missing values with statistical values, such as mean, median, and other statistical values, but this may lead to large imputation errors in some cases. Subsequently, similarity analysis (e.g., k-nearest neighbor (KNN) imputation [10]) and regression equation (e.g., multivariate imputation by chained equations (MICE) [11]) have been applied to the imputation of missing data, and these methods can achieve certain results with continuous development. However, when the data missing rate (MR) is relatively high, these methods may have large errors due to the reduction of effective information.

Manuscript received 6 June 2023; revised 25 July 2023; accepted 3 August 2023. Date of publication 23 August 2023; date of current version 1 September 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62303262, in part by the National Key Research and Development Program of China under Grant 2022YFF0610900, and in part by the China Postdoctoral Science Foundation Funded Project under Grant 2022TQ0179. The Associate Editor coordinating the review process was Dr. Yang Song. *(Corresponding author: Tao Sun.)*

Song Ma and Zeng-Song Xu are with the Key Laboratory of Intelligent Control and Optimization for Industrial Equipment, Ministry of Education, Dalian University of Technology, Dalian 116024, China (e-mail: masong@dlut.edu.cn; xuzengsong@mail.dlut.edu.cn).

Tao Sun is with the Department of Automation, Tsinghua University, Beijing 100084, China, and also with the Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116024, China (e-mail: suntao2022@tsinghua.org.cn).

Digital Object Identifier 10.1109/TIM.2023.3307765

As an active branch of artificial intelligence, neural networks have been used in the imputation area of missing data. Different from the above traditional imputation methods, the neural network imputation method generally includes training stage and imputation stage (i.e., first, a data-driven model with fixed parameters is obtained through model training, and then, the data generated by the trained model are used to impute the missing values). Early shallow neural network methods, such as multilayer perceptron (MLP) [12] and autoencoder (AE) [13], are weak in extracting MTS information due to their own structural limitations. With the improvement of hardware computing ability, deep learning method is becoming a hot topic in the imputation field of missing data because of its powerful information extraction capabilities. The existing deep learning imputation algorithms can be mainly divided into three categories: 1) gate recurrent unit (GRU)-based [14], [15], [16]; 2) generative adversarial network (GAN)-based [17], [18]; and 3) attention mechanism-based [19], [20]. As one of the most common modules in time-series missing value imputation, the GRU module is a recurrent neural network that introduces gating mechanisms to obtain the correlation of time series. However, the parallelism and information redundancy of the GRU module are poor, which can to some extent affect the effectiveness of data imputation. The method based on GANs predicts missing values from the distribution characteristics of observable data, enabling the estimated data to better fit the distribution of observable data. The method based on attention mechanism learns the temporal correlation of data from a global perspective, thus possessing good extraction ability for strong temporal data. In addition, due to the strong temporal and feature correlation of data from certain mechanical systems (such as aeroengines [2], [21]), it is a challenging task to estimate highly complex missing sensor data. Therefore, how to fully combine the advantages of existing methods and construct a data imputation method suitable for the field of industrial artificial intelligence has sparked our thinking.

In summary, the missing data imputation process of MTS involves the following issues: 1) how to establish a time-series model that considers data distribution and construct input data in a normative way, so as to utilize the temporal information of MTS? 2) how to introduce the multihead attention mechanism into the model to enhance the parallelism of the model and fully capture the temporal relationship information of MTS? and 3) how to improve the ability of the model to extract the feature correlation and make full use of the coupling between MTS? Motivated by these problems, we argue that high-quality data can accurately reflect the actual health status of the machine, and thus, it is very necessary to precisely impute the missing values in the dataset. Considering that the multisensor measurement data of some systems (such as aeroengines) have strong temporal correlation and feature correlation, it is a challenge to fully capture their spatiotemporal correlation. To the best of our knowledge, there is no existing work on MTS missing data imputation that considers data distribution, data temporality, and data feature correlation simultaneously. This article is based on the above discussion.

In this article, we will discuss a novel deep learning model for missing data imputation in MTS. The main contributions are as follows.

1) We propose a time-series generative adversarial imputation network (TSGAIN) framework for missing data imputation. The TSGAIN framework adopts a spatiotemporal structure, which fully retains the original time information of time series.
2) On the basis of the TSGAIN framework, we designed a TSGAIN with transformer encoder (TE-TSGAIN) and developed a TSGAIN with GRU module (GRU-TSGAIN) as the control group. TE-TSGAIN can mine the time correlation and distribution information of time series from a global perspective, which is superior to GRU-TSGAIN.
3) Based on the TE-TSGAIN model, we propose a TE-TSGAIN with preimputation and parallel convolution (PPCTE-TSGAIN). The preimputation module can provide more accurate input information for the model, and the parallel convolution (PC) module can enhance the model's ability to extract data feature correlations.
4) We conducted extensive experiments and ablation studies on the FD001, FD002, and FD003 subdatasets of the commercial modular aviation propulsion system simulation (C-MAPSS)[1] experimental dataset to quantitatively and qualitatively evaluate our method and justify its design. The results show that the PPCTE-TSGAIN not only achieves the highest imputation accuracy on the three subdatasets, but also demonstrates better adaptability to downstream tasks (RUL prediction of aeroengines).

The rest of this article is as follows. Section II introduces some necessary preliminary knowledge. Section III introduces the method of this article in detail, including problem formulation, data preparation, model structure, and specific implementation steps. Section IV dynamically evaluates the performance of the algorithm through experiments. Section V draws conclusions.

II. Preliminary Knowledge

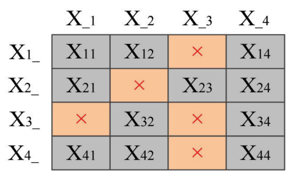
A. KNN Imputation and MICE

As two classic imputation methods, KNN imputation and MICE will be applied to some extent in preimputation process, so we will briefly introduce them. Assuming the input dataset is $X \in \mathbb{R}^{N\times F}$, i.e., there are N samples, each with F features. For ease of introduction, set $N = 4$ and $F = 4$, as shown in Fig. 1.

Here, $X_{1\_}$, $X_{2\_}$, $X_{3\_}$, and $X_{4\_}$ are different samples; $X_{\_1}$, $X_{\_2}$, $X_{\_3}$, and $X_{\_4}$ are different features of samples; for $\forall i \in \{1, 2, \ldots, N\}$, $\forall j \in \{1, 2, \ldots, F\}$, X_{ij} is the ith row and the jth column element of X; and X_{31}, X_{22}, X_{13}, X_{33}, and X_{34} are missing items in X.

1) KNN Imputation: The KNN imputation method finds K most similar samples for each missing value and fills in the missing value with the average of these K samples. First,

[1]https://www.kaggle.com/datasets/behrad3d/nasa-cmaps



	X_1	X_2	X_3	X_4
X1_	X11	X12	×	X14
X2_	X21	×	X23	X24
X3_	×	X32	×	X34
X4_	X41	X42	×	X44

Fig. 1. Input variable $X \in \mathbb{R}^{4\times 4}$.

for $\forall i, j \in \{1, 2, \ldots, N\}, i \neq j$, the similarity $d_{X_{i\_}, X_{j\_}}$ between two different samples is calculated according to the following process:

$$d_{X_{i\_}, X_{j\_}} = \sqrt{\frac{N}{N_{ij}^{\text{ob}}} \sum_{l=1}^{F} d_{X_{il}, X_{jl}}} \tag{1}$$

where N_{ij}^{ob} is the number of observable features for both $X_{i\_}$ and $X_{j\_}$, for example, $N_{12}^{\text{ob}} = 2$, which is because the first and fourth features in both $X_{1\_}$ and $X_{2\_}$ are observable, and $d_{X_{il}, X_{jl}}$ is the similarity of the lth feature between $X_{i\_}$ and $X_{j\_}$ shown as follows:

$$d_{X_{il}, X_{jl}} = \begin{cases} 0, & \text{if } X_{il} \text{ or } X_{jl} \text{ is missing} \\ (X_{il} - \hat{X}_{jl})^2, & \text{otherwise.} \end{cases} \tag{2}$$

Finally, the K samples that are most similar to the sample where a missing value is located are selected, and the average value is calculated to fill in the missing value.

2) MICE: MICE is a multiple imputation method for handling missing data, with the following steps [22].

1) *Initialization:* For each variable containing missing values, a simple method (such as mean, median, mode, and so on) is used to fill in the missing values and obtain an initial complete dataset, for example, for X_{31} in Fig. 1; X_{11}, X_{21}, and X_{41} are applied to simply fill in X_{31}.
2) *Iteration:* For each variable with missing values, do the following in order.
   a) *Delete:* Deletes all populated values for the variable from the current full dataset, restoring missing values. For example, for X_1 in Fig. 1, the simple imputed value of X_{31} is deleted.
   b) *Prediction:* Use other variables as predictors, establish a suitable imputation model (such as linear regression, logistic regression, polynomial regression, and so on), and predict the missing values of this variable. For example, for X_1 in Fig. 1, X_2, X_3, and X_4 are used as independent variables to predict the dependent variable X_1.
   c) *Imputation:* Updates the full dataset by extracting one or more values from the distribution of predicted values as filler values for that variable. For example, for X_1 in Fig. 1, the predicted value of X_{31} is used to fill in X_{31}.
3) *Repetition:* Repeat the iteration steps several times (such as 5, 10, and so on) to perform multiple imputations. After the last iteration, the complete dataset is output.

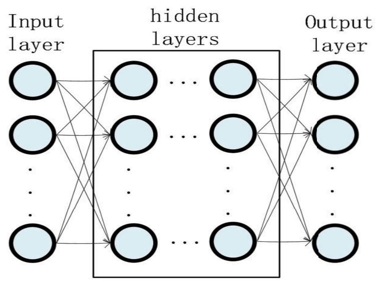


Fig. 2. Generator G and discriminator D of GAIN.

B. GAIN

GAIN [17] is a missing data imputation method using GANs [23]. GAIN is composed of a generator G and a discriminator D. Both the generator and discriminator structures are MLP, as shown in Fig. 2.

The input of the generator G is missing data vector $\mathbf{x} \in \mathbb{R}^F$, mask vector $\mathbf{m} \in \mathbb{R}^F$, and noise vector $\mathbf{z} \in \mathbb{R}^F$, where F is the number of features of $\mathbf{x}$, and the output of the generator is $G(\mathbf{x}, \mathbf{m}, \mathbf{z})$, which is used to impute missing values

$$\mathbf{x}^{\text{imp}} = \mathbf{x} \odot \mathbf{m} + G(\mathbf{x}, \mathbf{m}, \mathbf{z}) \odot (1 - \mathbf{m}) \tag{3}$$

where $\odot$ is the Hadamard product.

Next, GAIN designed a hint mechanism for the discriminator to enhance its recognition ability. The output of the discriminator is a probability vector

$$\mathbf{x}_D = D\left(\mathbf{x}^{\text{imp}}, \mathbf{h}\right) \tag{4}$$

where $\mathbf{h}$ is the hint vector and the element of $\mathbf{x}_D$ denotes the probability of corresponding element in $\mathbf{x}^{\text{imp}}$ comes from the original data.

The optimization objectives of GAIN are as follows:

$$\begin{aligned} &\min_D \mathcal{L}_D(\mathbf{m}, \mathbf{x}_D) \\ &\min_G \mathcal{L}_G\left(\mathbf{x}, \mathbf{m}, \mathbf{x}^{\text{imp}}, \mathbf{x}_D\right) \end{aligned} \tag{5}$$

where $\mathcal{L}_D$ is a cross-entropy loss function as follows:

$$\mathcal{L}_D(\mathbf{m}, \mathbf{x}_D) = -\mathbf{m} \log \mathbf{x}_D - (1 - \mathbf{m}) \log(1 - \mathbf{x}_D). \tag{6}$$

$\mathcal{L}_G$ is as follows:

$$\begin{aligned} &\mathcal{L}_G\left(\mathbf{x}, \mathbf{m}, \mathbf{x}^{\text{imp}}, \mathbf{x}_D\right) \\ &\quad = -(1 - \mathbf{m}) \log \mathbf{x}_D + \alpha \sum_{i=1}^{F} m_i \left(x_i - x_i^{\text{imp}}\right)^2 \end{aligned} \tag{7}$$

where m_i, x_i, and x_i^{imp} are the ith element of vectors $\mathbf{m}$, $\mathbf{x}$, and $\mathbf{x}^{\text{imp}}$, respectively, and α is a hyperparameter to balance two losses.

GAIN focuses on imputing nontemporal data, so its input data format is vector, and its generator and discriminator are relatively simple. Subsequently, the literature, such as [18], specifically improved GAIN and proposed PC-GAIN, which achieved good results. However, they did not pay more attention to the similarity of data in the time dimension, resulting in poor performance on some time-series datasets.

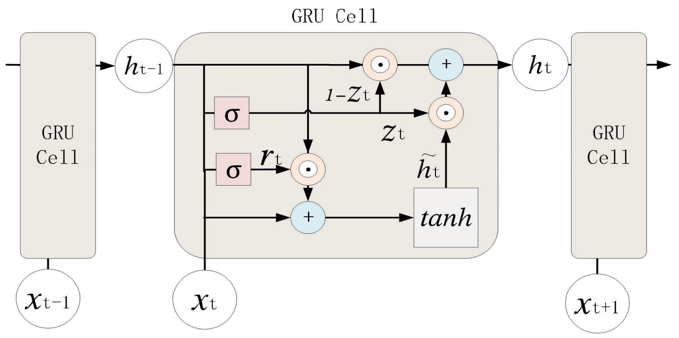


Fig. 3. Structure of GRU.

C. GRU

GRU is a kind of recurrent neural network that learns the correlation between time-series samples before and after to extract temporal information of a time-series system. A time-series sample can be described as $X = \{x_1, x_2, \ldots, x_N\}^T \in \mathbb{R}^{N\times F}$, where N is the number of time steps, F is the feature number, and for $\forall t \in \{1, 2, \ldots, N\}$, x_t is the sampling value of the sample at time t.

The structure of GRU is shown in Fig. 3, where the reset gate r_t is defined as follows:

$$r_t = \sigma\left(W_r\left[h_{t-1}, x_t\right]\right) \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid activation function, h_{t-1} is the output from the previous time, x_t is the input of the current time, and W_r is a trainable parameter vector for reset gate.

The update gate z_t is defined as follows:

$$z_t = \sigma\left(W_z\left[h_{t-1}, x_t\right]\right) \tag{9}$$

where W_z is a trainable parameter vector for update gate.

Next, the preoutput $\tilde{h}_t$ at the current time point is calculated as follows:

$$\tilde{h}_t = \tanh\left(W_{\tilde{h}} \cdot \left[r_t \odot h_{t-1}, x_t\right]\right) \tag{10}$$

where $\tanh(\cdot)$ is a kind of activation function and $W_{\tilde{h}}$ is a trainable parameter vector.

Finally, the update gate z_t is used to weigh the previous output h_{t-1} and preoutput $\tilde{h}_t$ to obtain the output h_t at this time

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \tag{11}$$

In fact, GRU modules have made significant contributions in the field of data imputation [14], [15], [16], but with the development of deep learning, an attention mechanisms-based temporal learning method called transformer has become a trend in many deep learning fields [24], [25]. Therefore, attention mechanisms will be introduced in our work, and then, the GRU, which is currently the mainstream method in the field of temporal data imputation, will be introduced as the control group.

III. Theoretical Methods

In this section, a TSGAIN with preimputation, PC, and transformer encoder (PPCTE) will be proposed. The construction of the model includes five parts as follows.

A. Problem Formulation

In general, the time series can be described as a state vector composed of measured values at different time steps. Then, the state vectors of multiple sensors can be spliced into a second-order state tensor, which mainly contains time and feature (i.e., sensor) information. In fact, due to the influence of external factors, such as electromagnetic interference and voltage instability, some values may be lost during sensor measurement, which are nonnumeric in the state matrix.

Specifically, suppose that the research target has F sensors, and the number of time steps measured by each sensor is N. Then, the raw data object can be described as a state matrix

$$X^{\text{raw}} = \left[X_1^{\text{raw}}, X_2^{\text{raw}}, \ldots, X_f^{\text{raw}}, \ldots, X_F^{\text{raw}}\right] \in \mathbb{R}^{N\times F} \tag{12}$$

where for $\forall f \in \{1, 2, \ldots, F\}$, $\forall n \in \{1, 2, \ldots, N\}$, $X_f^{\text{raw}} = [X_{1f}^{\text{raw}}, X_{2f}^{\text{raw}}, \ldots, X_{nf}^{\text{raw}}, \ldots, X_{Nf}^{\text{raw}}]^T \in \mathbb{R}^{N\times 1}$ is the time-series data of the fth sensor and X_{nf}^{raw} is the value of the fth sensor at the nth time step.

If the matrix element is missing, we use the symbol NAN to represent the missing value. For example, when $F = 5$ and $N = 3$, the state matrix is as follows:

$$X^{\text{raw}} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ \text{NAN} & 9 & \text{NAN} & \text{NAN} & 6 \\ 24 & \text{NAN} & 30 & 8 & 0 \end{bmatrix}$$

where $X_1^{\text{raw}} = [2, \text{NAN}, 24]^T$, $X_2^{\text{raw}} = [0, 9, \text{NAN}]^T$, $X_3^{\text{raw}} = [0, \text{NAN}, 30]^T$, $X_4^{\text{raw}} = [0, \text{NAN}, 8]^T$, and $X_5^{\text{raw}} = [0, 6, 0]^T$ are the time series of the first, second, third, fourth, and fifth sensors, respectively.

B. Data Preparation

1) Marking Missing Values: The missing values in the dataset have been marked by the symbol NAN, but NAN (a nonnumeric symbol) cannot be used for model training. To mark and distinguish the missing and nonmissing values in the state matrix $X^{\text{raw}} \in \mathbb{R}^{N\times F}$, we introduce a definition process of a raw mask matrix $M^{\text{raw}} \in \mathbb{R}^{N\times F}$ as shown below

$$M_{nf}^{\text{raw}} = \begin{cases} 0, & \text{if } X_{nf}^{\text{raw}} = \text{NAN} \\ 1, & \text{otherwise} \end{cases} \tag{13}$$

where for $\forall f \in \{1, 2, \ldots, F\}$, $\forall n \in \{1, 2, \ldots, N\}$, M_{nf}^{raw} is the nth row and the fth column element of M^{raw}.

Specially, the MR of multivariate missing time series can be defined as follows:

$$\text{MR} = \frac{\sum_{n=1}^{N}\sum_{f=1}^{F}\left(1 - M_{nf}^{\text{raw}}\right)}{N \times F}. \tag{14}$$

2) Data Standardization: The data measured by different sensors generally have different scales, which will adversely affect the training process of the model. Therefore, standardization method can be used to make different sensor data have the same scale. For state matrix $X^{\text{raw}} \in \mathbb{R}^{N\times F}$, the standardization process is defined as follows:

$$X_{nf}^{\text{norm}} = \begin{cases} \dfrac{X_{nf}^{\text{raw}} - \mu_f}{\sigma_f}, & \text{if } M_{nf}^{\text{raw}} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

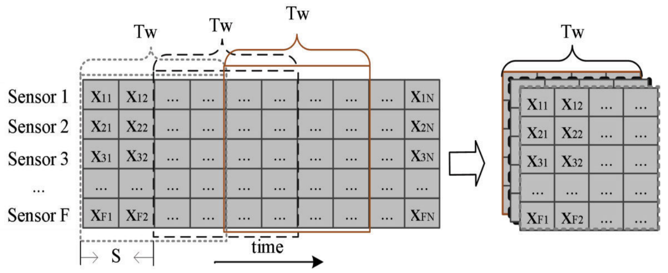


Fig. 4. Diagram of the sliding time window method.

where for $\forall f \in \{1, 2, \ldots, F\}$, $\forall n \in \{1, 2, \ldots, N\}$, X_{nf}^{norm} is the nth row and the fth column element of the standardized matrix $X^{\text{norm}} \in \mathbb{R}^{N\times F}$, and $\mu_f \in \mathbb{R}$ and $\sigma_f \in \mathbb{R}$ are the average and variance value of nonmissing elements of the fth sensor, respectively.

3) Sliding Time Window Method: In order to fully retain the temporal information in $X^{\text{norm}} \in \mathbb{R}^{N\times F}$ and $M^{\text{raw}} \in \mathbb{R}^{N\times F}$, the sliding time window method is used to construct the input samples. For the transposition of standardized matrix: $(X^{\text{norm}})^T \in \mathbb{R}^{F\times N}$, the specific description of sliding time window method is shown in Fig. 4.

Specifically, we slide orderly in the time direction of $(X^{\text{norm}})^T \in \mathbb{R}^{F\times N}$ and finally generate the subsequent input $X \in \mathbb{R}^{\text{num}\times(Tw\times F)}$, where X is the input sample corresponding to the standardized matrix X^{norm}, num is the number of samples with size $Tw \times F$, Tw is the time window size, and S in Fig. 4 is the interval between two adjacent windows.

Similarly, for the transposition of the raw mask matrix: $(M^{\text{raw}})^T \in \mathbb{R}^{F\times N}$, $M \in \mathbb{R}^{\text{num}\times(Tw\times F)}$ is finally obtained by the same method as X, where M is the input sample corresponding to the raw mask matrix M^{raw}.

4) Preimputation: The missing elements (i.e., unobservable information) will be reconstructed by constructing a reasonable model related to nonmissing elements (i.e., observational information). The reasonable utilization of observable information is extremely important. As a necessary condition for building the proposed model, KNN imputation or MICE imputation will be used for preimputation, which further provides more accurate information for the input of the proposed model. Specifically, we utilize KNN imputation or MICE imputation to conduct information mining for X in advance and finally get the output $X_{\text{pre}} \in \mathbb{R}^{\text{num}\times(Tw\times F)}$, which is one of the input samples of the proposed model.

C. Proposed Framework

The main framework of the proposed PPCTE-TSGAIN model is shown in Fig. 5, and the most important parts of the model are the generator and the discriminator. For the convenience of discussion, we assume that the batch size of the input sample is 1; then, the model input samples can be described as $X \in \mathbb{R}^{Tw\times F}$, $M \in \mathbb{R}^{Tw\times F}$, and $X_{\text{pre}} \in \mathbb{R}^{Tw\times F}$.

In general, it can be seen from Fig. 5 that the generator performs a regression task, which is to generate data to impute the missing elements. Specifically, before X is input into the generator, Gaussian noise with an average value of 0 and a

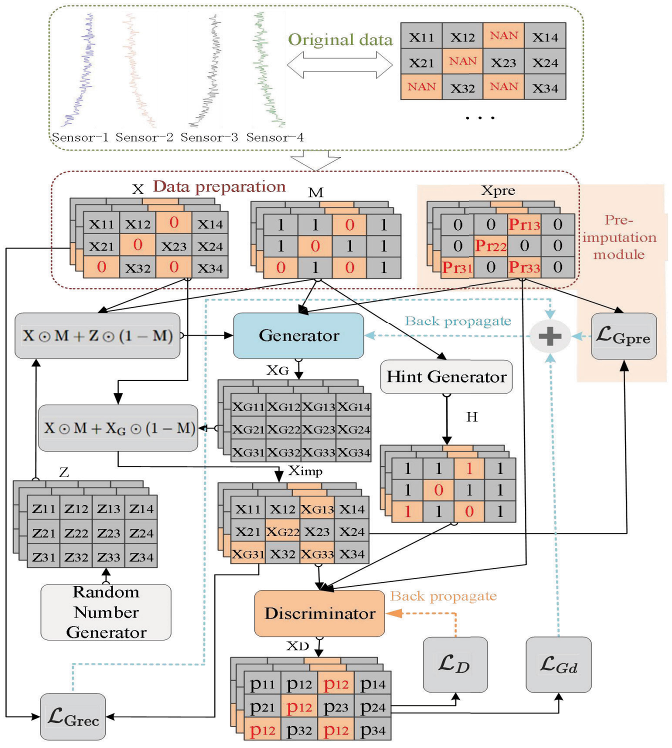


Fig. 5. Overview of the proposed PPCTE-TSGAIN.

variance of 0.01 will be generated by the random number generator, and then, the Gaussian noise will be added to the missing elements of X. Next, X, M, and X_{pre} are input into the generator; then, the generator finally generates a numeric matrix: $X_G \in \mathbb{R}^{Tw\times F}$.

Next, the discriminator performs a classification task, which is to distinguish "true" and "false" data (in the imputation process, the nonmissing data are "true," and the imputed data are "false"). Specifically, first, the imputed matrix $X_{\text{imp}} \in \mathbb{R}^{Tw\times F}$ will be obtained by the following process:

$$X_{\text{imp}} = X \odot M + X_G \odot (1 - M) \tag{16}$$

where $\odot$ is the Hadamard product. Then, the hint generator similar to the literature [17] will generate the hint matrix $H \in \mathbb{R}^{Tw\times m}$, which randomly sets the elements with the value of 0 in M to 1 according to a certain probability Hrate (here, Hrate $= 0.9$). Next, X_{imp}, H, and X_{pre} are input into the discriminator, and finally, the discriminator outputs a probability matrix: $X_D \in \mathbb{R}^{Tw\times F}$. Specially, for $\forall n \in \{1, 2\ldots, Tw\}$, $\forall f \in \{1, 2\ldots, F\}$, the nth row and the fth column element of X_D is $(X_D)_{nf}$, which represents the probability that the corresponding $(X_{\text{imp}})_{nf}$ in X_{imp} is "true" to distinguish between imputed data and nonmissing data.

The detailed structure of the generator and discriminator and the training process of the proposed PPCTE-GAIN model will be discussed in Sections III-D and III-E, respectively.

D. Generator and Discriminator Networks

The generator and discriminator are mainly composed of PC $\rightarrow$ U $\times$ transformer encoder $\rightarrow$ PC (PCTE) shown in Fig. 6. Specifically, PCTE is an encoder–decoder architecture.

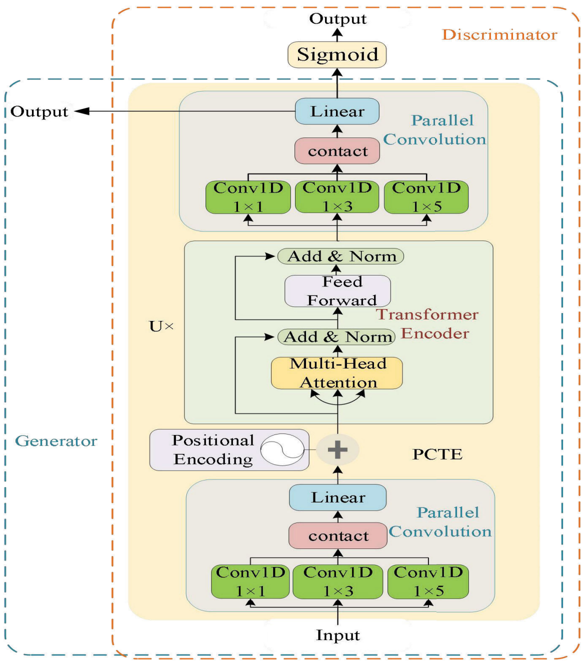


Fig. 6. Structure of generator and discriminator.

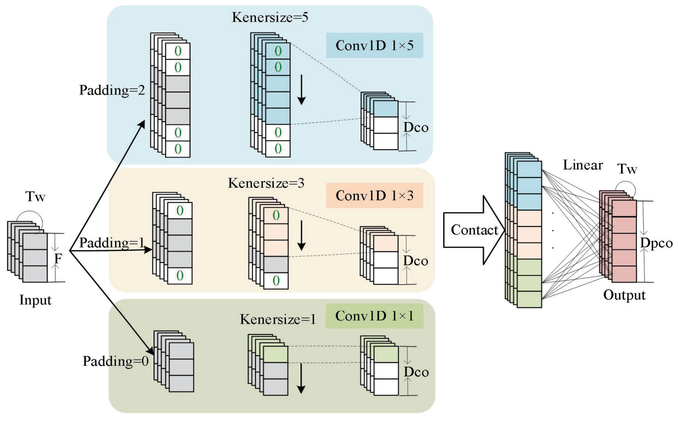


Fig. 7. Structure of the proposed PC.

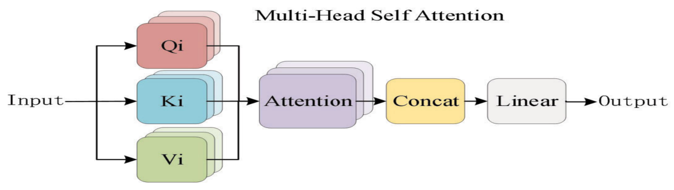


Fig. 8. Structure of the MHSA.

Encoder: First, the first PC module extracts feature correlation from the original dimension data. Second, the data feature dimension is enlarged to increase the expressiveness of the feature. Third, the positional encoding (PE) module adds the temporal information to the data. Fourth, the time information of high feature dimension data is extracted from the state extraction layer stacked by the U transformer encoders (TEs). Finally, the second PC module extracts the feature correlation from high-dimensional data. Through the above modules, the internal correlation information of the data is fully extracted.

Decoder: For regression task similar to the generator task, the linear layer of the second PC module can directly serve as the decoder to achieve data dimensionality reduction. For the classification task similar to the discriminator task, the decoding can be completed by adding the required module (such as the sigmoid module required by the discriminator) at the end.

Next, we will discuss relevant components in detail. For the convenience of discussion, we assume that the input data are $\alpha \in \mathbb{R}^{Tw\times F}$ (actually, the input data are the splicing of X, M, and $X_{\text{pre}} \in \mathbb{R}^{Tw\times 3F}$).

1) Parallel Convolution: The PC module consists of a Conv1D [26] parallel layer with different convolution cores, a contact layer, and a linear layer, where the convolution kernel size of Conv1D$_{1\times 1}$ is 1, that of Conv1D$_{1\times 3}$ is 3, and that of Conv1D$_{1\times 5}$ is 5, and its structure is shown in Fig. 7.

Specifically, $\alpha \in \mathbb{R}^{Tw\times F}$ is input to the Conv1D parallel layers; then, for $\forall i \in \{1, 3, 5\}$, the output Conv1D$_{1\times i}(\alpha) \in \mathbb{R}^{Tw\times \text{Dco}}$, where $D_{\text{co}} = F - \text{Kernelsize} + 2 \times \text{Padding} + 1$ is the feature dimension of the output of Conv1D$_{1\times i}$ modules, Kernelsize is the size of convolution kernel, Padding is the number of zeros on one side (we will control padding to ensure $D_{\text{co}} = F$), and Tw is the time dimension, which will remain unchanged by keeping the size of the input channel and output channel of the Conv1D$_{1\times i}$ modules equal.

Then, the contact layer splices the Conv1D$_{1\times i}(\alpha)$ modules by feature dimension. Finally, the linear layer automatically learns the weight $W^P \in \mathbb{R}^{3D_{\text{co}}\times D_{\text{pco}}}$ for weighted summation, and output $Y_{\text{pc}} \in \mathbb{R}^{Tw\times D_{\text{pco}}}$, where Y_{pc} is also the output of the PC module. Therefore, by increasing or reducing D_{pco}, we can increase or decrease the feature dimension of the data.

2) Positional Encoding: Since the attention module must learn the temporal information of the data according to some additional prompts, we add a PE module to inject position information into the time series, so that the model can utilize the temporal information of the time series. For the output $Y_{\text{pc}} \in \mathbb{R}^{Tw\times D_{\text{pco}}}$ of the PC module, the definition process of position matrix $\text{PE} \in \mathbb{R}^{Tw\times D_{\text{pco}}}$ is as follows:

$$\begin{aligned} \text{PE}_{(i,2j)} &= \sin(i/10000^{2j/D_{\text{pco}}}) \\ \text{PE}_{(i,2j+1)} &= \cos(i/10000^{2j/D_{\text{pco}}}) \end{aligned} \tag{17}$$

where for $\forall i \in \{1, 2 \ldots Tw\}$, $\forall j \in \{0, 1 \ldots \lfloor D_{\text{pco}}/2 \rfloor\}$ ($\lfloor\cdot\rfloor$ indicates rounding down), $\text{PE}_{(i,2j)}$ is the element of the ith row and the $2j$th column of the PE matrix, and $\text{PE}_{(i,2j+1)}$ is the element of the ith row and the $(2j+1)$th column of the PE matrix.

Each dimension encoded by this position corresponds to a different sine or cosine curve, and the position of the input data can be uniquely marked. Finally, we use $Y_{\text{pe}} = Y_{\text{pc}} + \text{PE}$ to embed the position coding information into Y_{pc} to get the output of PE module $Y_{\text{pe}} \in \mathbb{R}^{Tw\times D_{\text{pco}}}$.

3) Transformer Encoder: TE [27] is a powerful module for extracting temporal correlation, and we will introduce it in detail.

a) Multihead self-attention: The multihead self-attention (MHSA) is shown in Fig. 8, and we take output $Y_{\text{pe}} \in \mathbb{R}^{Tw\times D_{\text{pco}}}$ of PE module as the input. First, $Y_{\text{pe}} \in \mathbb{R}^{Tw\times D_{\text{pco}}}$ is mapped to three submatrices, which are query matrix $Q \in \mathbb{R}^{Tw\times D_{\text{pco}}}$, key matrix $K \in \mathbb{R}^{Tw \times D_{\text{pco}}}$, and value matrix $V \in \mathbb{R}^{Tw \times D_{\text{pco}}}$, respectively, as shown below

$$Q = Y_{\text{pe}} W_Q, \quad K = Y_{\text{pe}} W_K, \quad V = Y_{\text{pe}} W_V \tag{18}$$

where W_Q, W_K, $W_V \in \mathbb{R}^{D_{\text{pco}} \times D_{\text{pco}}}$ are learnable matrices.

Then, Q, K, and V are equally divided into h attention heads in the feature dimension

$$\begin{aligned} &Q = [Q_1, Q_2, \dots, Q_h], \quad K = [K_1, K_2, \dots, K_h] \\ &V = [V_1, V_2, \dots, V_h] \end{aligned} \tag{19}$$

where for $\forall i \in \{1, 2, \dots, h\}$, $\{Q_i, K_i, V_i \in \mathbb{R}^{Tw \times (D_{\text{pco}}/h)}\}$ is the ith attention head.

Then, we calculate the attention score through the dot product of matrix Q_i and matrix K_i. Next, the attention score is multiplied by $(1/(D_{\text{pco}})^{1/2})$. Then, attention is pooled through softmax operation, which make the sum of the attention score be 1. Finally, we calculate the dot product of the attention score and matrix V_i, so that the network can assign different weights to different channels according to the attention score

$$\text{Attention}(Q_i, K_i, V_i) = \text{Softmax}\left(\frac{Q_i {K_i}^T}{\sqrt{D_{\text{pco}}}}\right) Vi. \tag{20}$$

Finally, the attention results of different heads are spliced into the final result matrix $Y_{\text{MHSA}} \in \mathbb{R}^{Tw \times D_{\text{pco}}}$

$$Y_{\text{MHSA}} = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O \tag{21}$$

where $\text{head}_i = \text{Attention}(Q_i, K_i, V_i)$ and $W^O \in \mathbb{R}^{D_{\text{pco}} \times D_{\text{pco}}}$ is learnable matrices. By splicing h centralized outputs, the multihead structure not only has more parallelism, but also focuses on the coding information contained in more subspaces.

b) Transformer encoder: The TE module shown in Fig. 6 is based on the above MHSA module. In addition, the TE module also contains some simple modules, such as add and norm and feedforward, and none of them will change the dimension of the input data, so we use $z \in \mathbb{R}^{Tw \times Dz}$ to represent the input for a brief introduction, where Tw and Dz are the time and feature dimensions of z, respectively.

The add and norm module contains residual connections and layer standardization as follows:

$$Y_{\text{an}}(z) = \text{layernorm}(\text{sublayer}(z) + z) \tag{22}$$

where $Y_{\text{an}}(z) \in \mathbb{R}^{Tw \times Dz}$ denotes the output of the add and norm module, sublayer$(\cdot)$ is the output of the previous layer, and layernorm$(\cdot)$ denotes the layer normalization.

For $z \in \mathbb{R}^{Tw \times Dz}$, the feedforward module is a two-layer network module, as shown below

$$Y_{\text{ff}} = \max(0, zW_1 + b_1)W_2 + b_2 \tag{23}$$

where $Y_{\text{ff}} \in \mathbb{R}^{Tw \times Dz}$ is the output of the feedforward module, $W_1 \in \mathbb{R}^{Dz \times Dz}$ and $W_2 \in \mathbb{R}^{Dz \times Dz}$ represent the weight of the corresponding layer, $b_1 \in \mathbb{R}^{Tw \times Dz}$ and $b_2 \in \mathbb{R}^{Tw \times Dz}$ represent the offset of the corresponding layer, and they are all learnable parameters.

The outputs of the MHSA module, the add and norm module, and the feedforward module all do not change the dimension of input, so the TE module can be U-stacked to fully mine the data information.

E. Procedure

The training process for PPCTE-TSGAIN is described in Algorithm 1, and it can also be described as the following optimization process:

$$\begin{aligned} &\min_D \frac{1}{\text{BS}} \sum_{k=1}^{\text{BS}} \mathcal{L}_D^k \\ &\min_G \frac{1}{\text{BS}} \sum_{k=1}^{\text{BS}} \mathcal{L}_G^k \end{aligned} \tag{24}$$

where G and D are the generator and discriminator, respectively; BS is the number of samples input per time; $\mathcal{L}_G$ and $\mathcal{L}_D$ are the loss function of G and D, respectively; and for $\forall k \in \{1, 2, \dots, \text{BS}\}$, $\mathcal{L}_G^k$ and $\mathcal{L}_D^k$ are the values of $\mathcal{L}_G$ and $\mathcal{L}_D$ of the kth sample, respectively.

First, we will discuss the optimization process of D. First, Gaussian noise with mean 0 and variance 0.01 is added to the missing elements of X. Second, X, M, and X_{pre} are spliced and input to G to obtain the output X_G. Third, according to $X_{\text{imp}} = X \odot M + X_G \odot (1 - M)$, X_G is used to impute X, and then, the imputed matrix X_{imp} is obtained, and M is input to hint generator to obtain H. Fourth, X_{imp}, H, and X_{pre} are spliced as the input of D to obtain output X_D. Finally, for $\forall k \in \{1, 2, \dots, \text{BS}\}$, the D loss $\mathcal{L}_D^k$ in formula (24) can be obtained by the same process

$$\begin{aligned} \mathcal{L}_D(M, X_D) = -\frac{1}{Tw \times F} \sum_{i=1}^{Tw} \sum_{j=1}^{F} \big[&M_{ij} \log{(X_D)_{ij}} + (1 - M_{ij}) \\ &\times \log(1 - (X_D)_{ij}) \big]. \end{aligned} \tag{25}$$

Then, the D loss is used for backpropagation to update generator parameters θ_D.

Then, the generator is optimized. The acquisition step of X_D is the same as that of the discriminator (but it has to be retrieved, because the discriminator has been updated); for $\forall k \in \{1, 2, \dots, \text{BS}\}$, the G loss $\mathcal{L}_G^k$ in formula (24) can be obtained by the same process

$$\mathcal{L}_G(X, X_{\text{pre}}, M, X_D) = \mathcal{L}_{Gd} + \lambda \mathcal{L}_{G\text{rec}} + \beta \mathcal{L}_{G\text{pre}} \tag{26}$$

where $\mathcal{L}_{Gd}$ is the cross-entropy loss function from the discriminator shown as follows:

$$\mathcal{L}_{Gd}(M, X_D) = \frac{-1}{Tw \times F} \sum_{i=1}^{Tw} \sum_{j=1}^{F} (1 - M_{ij}) \log(X_D)_{ij}. \tag{27}$$

$\mathcal{L}_{G\text{rec}}$ is the reconstruction loss function of X as shown in the following, respectively:

$$\mathcal{L}_{G\text{rec}}(X, M, X_G) = \|X \odot M - X_G \odot M\|_2. \tag{28}$$

$\mathcal{L}_{G\text{pre}}$ is the reconstruction loss function of X_{pre} as shown in the following:

$$\mathcal{L}_{G\text{pre}}(X_{\text{pre}}, M, X_G) = \big\| X_{\text{pre}} \odot (1 - M) - X_G \odot (1 - M) \big\|_2. \tag{29}$$

λ and β are to balance the values between the three losses. Then, the G loss is used for backpropagation to update generator parameters θ_G.

Algorithm 1 Training Stage of the PPCTE_TSGAIN Model

```
Input:
    Current batch of training data set, X, M and $X_{pre}$.
    Number of epoch.
    Hyper-parameter $\lambda$, $\beta$.
    Initialized parameters: Generator-$\theta_G$ and Discriminator-$\theta_D$.
for i = (1, 2, ..., epoch)
  1) Discriminator optimization
    1. $X \leftarrow X \odot M + Z \odot (1 - M)$.
    2. $X_G \leftarrow G(\mathrm{Contact}(X, M, X_{pre}))$.
    3. $X_{imp} \leftarrow X \odot M + X_G \odot (1 - M)$, $H \leftarrow \mathrm{Hint}(M)$.
    4. $X_D \leftarrow D(\mathrm{Contact}(X_{imp}, H, X_{pre}))$.
    5. D loss $\leftarrow$ formula (25).
    Back-propagate D loss to update $\theta_D$.
  2) Generator optimization
    1. $X_G \leftarrow G(\mathrm{Contact}(X, M, X_{pre}))$.
    2. $X_{imp} \leftarrow X \odot M + X_G \odot (1 - M)$, $H \leftarrow \mathrm{Hint}(M)$.
    3. $X_D \leftarrow D(\mathrm{Contact}(X_{imp}, H, X_{pre}))$.
    4. $\mathcal{L}_{Gd} \leftarrow$ formula (27).
    5. $\mathcal{L}_{Grec} \leftarrow$ formula (28).
    6. $\mathcal{L}_{Gpre} \leftarrow$ formula (29).
    7. G loss $\leftarrow$ formula (26).
    Back-propagate G loss to update $\theta_G$.
end for
Output:
  Trained parameters Generator-$\theta_G$.
```

In the training stage, the discriminator and generator are trained alternately. Finally, an end-to-end generator model with trained parameters is obtained at the end of training. We can input the incomplete dataset into the generator model and then generate the MTS X_G to impute the incomplete dataset by $X_{\mathrm{imp}} = X \odot M + X_G \odot (1 - M)$.

IV. Simulation Experiments

In this section, we will conduct experiments on the C-MAPSS data to verify the effectiveness of the proposed model. The experimental procedure mainly consists of a training stage and a test stage shown in Fig. 9, where the image of the aeroengine in the flowchart is sourced from the website.[2]

A. C-MAPSS Dataset

The simulation dataset in this article is the C-MAPSS dataset,[3] which is a publicly available dataset commonly used for aeroengine health management. The C-MAPSS dataset has four subsets FD001, FD002, FD003, and FD004, where FD001, FD002, and FD003 are selected as the experimental datasets. In each subdataset, there are data of multiple aeroengines, and each aeroengine has 21 sensors, as shown in Table I. It is worth noting that the data of sensors 1–9 are directly measured, including temperature sensors 1–4, pressure sensors 5–7, and speed sensors 8 and 9, while

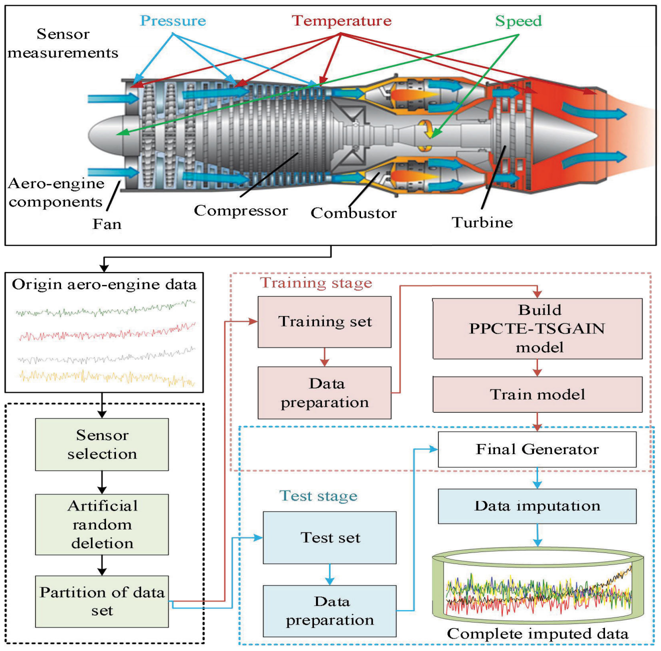


Fig. 9. General overview of the experimental process.

TABLE I
Meaning of Each Sensor in the Dataset

No.	Symbol	Description	Units
1	T2	Total temperature at fan inlet	°R
2	T24	Total temperature at LPC outlet	°R
3	T30	Total temperature at HPC outlet	°R
4	T50	Total temperature at LPT outlet	°R
5	P2	Pressure at fan inlet	psia
6	P15	Total pressure in bypass-duct	psia
7	P30	Total pressure at HPC outlet	psia
8	Nf	Physical fan speed	rpm
9	Nc	Physical core speed	rpm
10	epr	Engine pressure ratio (P50/P2)	–
11	Ps30	Static pressure at HPC outlet	psia
12	phi	Ratio of fuel flow to Ps30	pps/psi
13	NRf	Corrected fan speed	rpm
14	NRc	Corrected core speed	rpm
15	BPR	Bypass Ratio	–
16	farB	Burner fuel-air ratio	–
17	htBleed	Bleed Enthalpy	–
18	Nf_dmd	Demanded fan speed	rpm
19	PCNfR_dmd	Demanded corrected fan speed	rpm
20	W31	HPT coolant bleed	lbm/s
21	W32	LPT coolant bleed	lbm/s

some sensor data are indirectly obtained based on a combination of measurement and empirical knowledge, including sensors 10–21.

However, the values of some sensors will not change throughout their life cycle, and these sensor data are not useful for subsequent analysis. Therefore, we perform the analysis of correlations between features in a similar way to the literature [3], eventually eliminating sensors that are not useful. Specifically, for example, the trend of FD001 sensor over time cycles (i.e., time steps) is shown in Fig. 10, where for $\forall i \in \{1, 2, \ldots, 21\}$, s_i is the ith sensor data of all engines in FD001. It can be seen that the sensor values will not change with time, so we eliminate sensor numbers 1, 5, 6, 10, 16, 18,

[2]https://zh.wikipedia.org/wiki/File:Jet_engine.svg
[3]https://www.kaggle.com/datasets/behrad3d/nasa-cmaps

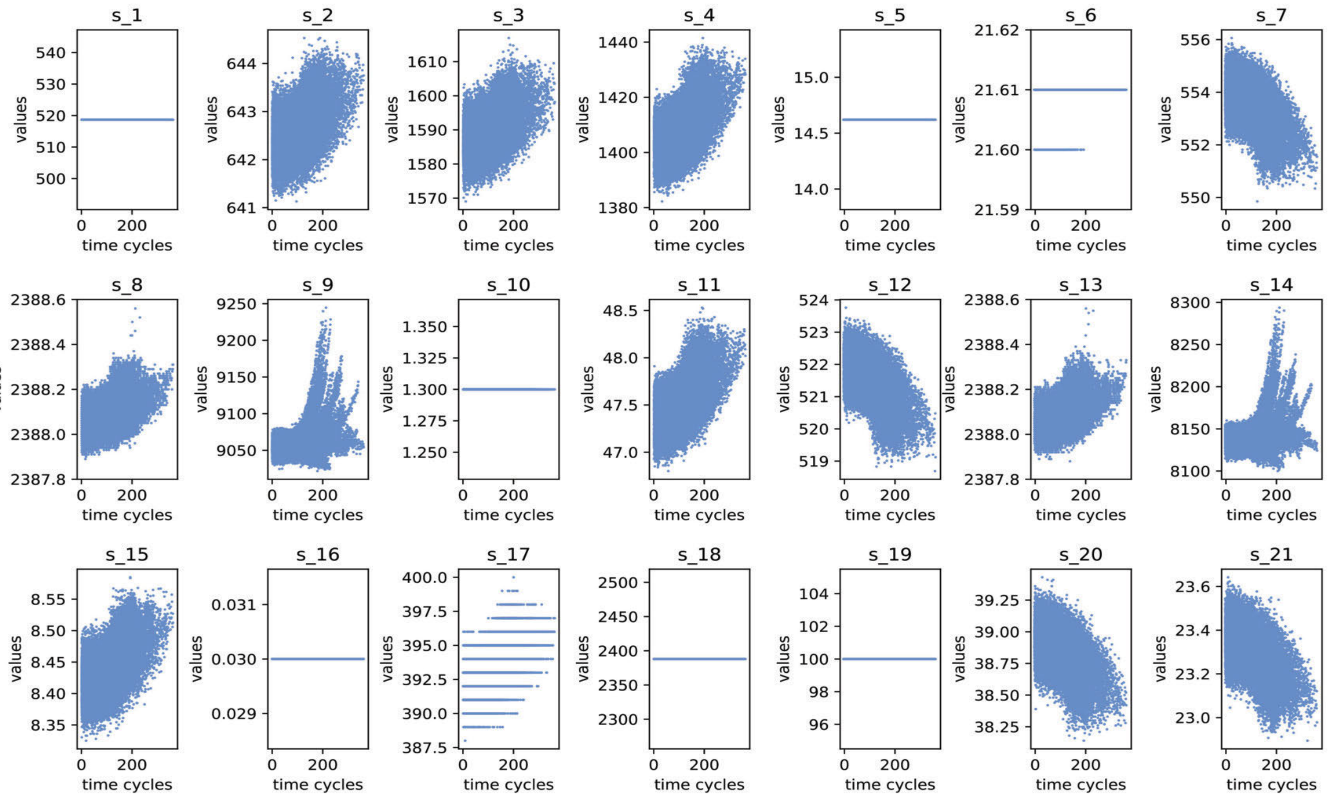


Fig. 10. Correlation of all engine sensor data to time cycles in FD001, where for $\forall i \in \{1, 2, \ldots, 21\}$, s_i is the ith sensor data of all engines in FD001.

TABLE II
NUMBER OF ENGINES IN THE TRAINING AND TEST SETS AND THE NUMBER OF EFFECTIVE SENSORS

Subsets	FD001	FD002	FD003
Number of training engines	160	415	160
Number of testing engines	40	104	40
Effective sensors	14	21	16
Conditions	1	6	1
Fault modes	1	1	2

and 19 in FD001. Similarly, we preserve all sensors in FD002 and eliminate sensor numbers 1, 5, 16, 18, and 19 in FD003.

In order to fully evaluate the robustness of the model, we set up simulation experiments under multiple MR conditions; that is, according to the MR defined by formula (14), we set nine MR conditions: {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}. Then, the data are artificially randomly missing at the MR condition.

Finally, we use the out-of-sample method to divide the dataset, that is, randomly scramble the engine number as the minimum unit, select 80% engines as the training set, and 20% engines as the test set for simulation verification, as shown in Table II.

B. Imputation Tasks

Our main task is to impute the missing data. Because the data missing is simulated artificially, we can retain the true values of the missing elements to calculate the overall difference between the imputed value and the true value, so as to evaluate the imputation result, i.e., we use RMSE $= ((1/n)\sum_{i=1}^{n}(y_i - \hat{y}_i)^2)^{1/2}$ to evaluate the imputation performance, where n is the total number of missing elements, y_i is the true value, $\hat{y}_i$ is the imputed value, and the smaller the root mean square error (RMSE), the smaller the difference between the imputed values and the true values, and the better the imputation performance. In addition, it is worth noting that in order to eliminate the impact of different feature scales, we will use the standardized data to evaluate the imputation performance.

C. Baseline Methods

1) Statistical Value Imputation Method: Statistical value imputation is to use certain statistical values to impute the missing data, where mean imputation and median imputation are used as baseline methods in this article. *KNN imputation*: KNN imputation [10] is an improvement of the KNN clustering method, which is first to find the closest K samples in the data space and then estimate the value of the missing vales by the K samples. *MICE*: MICE [11] uses the idea of regression to build regression models for nonmissing values and to make predictions for missing values. *GAIN*: GAIN[4] [17] is a method for missing data imputation using GANs, mainly for nonsequential data. *GRU-TSGAIN*: GRU-TSGAIN based on the idea of [28], using GRU module and GAN frame for time-series missing data imputation. *Brits*: Brits[5] [14] is a imputation model for time series, mainly using bidirectional GRU model.

[4]https://github.com/jsyoon0823/GAIN
[5]https://github.com/caow13/Brits

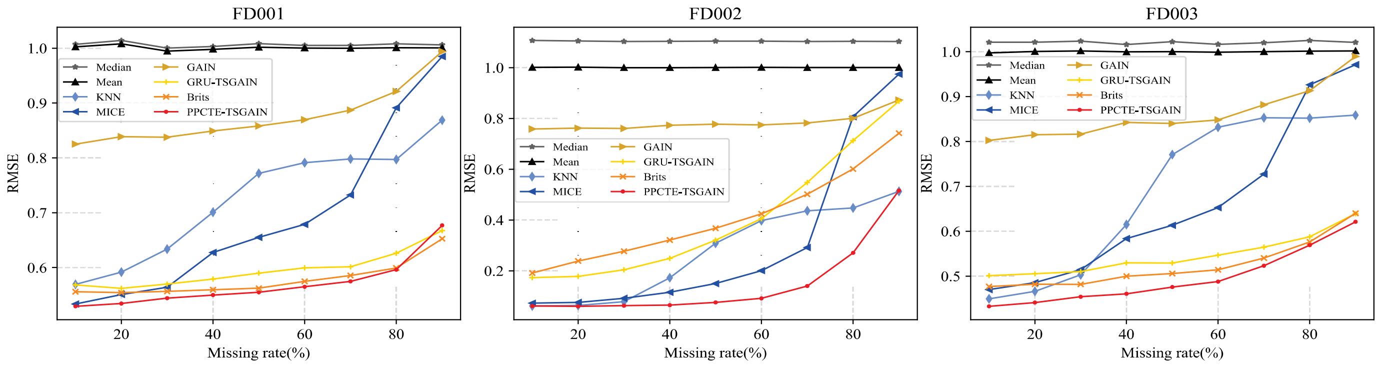


Fig. 11. Performance comparison for imputation on FD001, FD002, and FD003. The horizontal axis is the MR, and the vertical axis is the RMSE of standardized data.

D. Specific Realization

We implement the program in a 64-bit Windows 11 desktop environment with 16G RAM, Intel i5-12500H CPU, and NVIDIA GeForce RTX 3060 Laptop GPU, and we use Python 3.8.13 as the programming language and PyTorch 1.11.0 as the deep learning framework.

The statistical value imputation, KNN imputation [10], and MICE imputation [11] are implemented by calling Sklearn Imputer [29].[6] Specifically, the statistical value imputation is implemented by calling SimpleImputer, the KNN imputation is implemented by calling KNNImputer with n_neighbors = 10, and MICE imputation is implemented by calling IterativeImputer with max_iter = 5.

GAIN [17] is a deep learning imputation model for nonsequential data, and we follow its original data input format by directly disrupting the original dataset and validating it in the form of 80% training set and 20% test set.

GRU-TSGAIN, Brits [14], and PPCTE-TSGAIN are imputation models for time series, and their data formats are the same. To compare each model fairly, the Tw of the sliding window method for data processing is all 30, the step size S is all 5, and the batch size of the input data is all 128. In addition, the important parameters of various models are dynamically optimized; that is, the best value in a certain interval is found; for example, the λ of PPCTE-TSGAIN is optimized in the set $\{0.1, 1, 10, 100\}$, and finally, 1 is selected.

Specifically, for the proposed PPCTE-TSGAIN model, KNN imputation is chosen for the preimputation module of FD001 and FD003, while MICE imputation is chosen for the preimputation module of FD002 (since the amount of data in FD002 is too large, and KNN imputation is time-consuming). The important parameters of the PPCTE-TSGAIN model are shown below, for FD001 and FD003: the number of the TE heads $h = 8$, the number of stacking layers $U = 2$, the number of hidden layers (i.e., the D_{pco} of the first PC module) is 128, epochs = 20, $\lambda = 1$, $\beta = 0.1$, the generator learning rate $\text{lr}_G = 3e^{-5}$, the discriminator learning rate $\text{lr}_D = 6e^{-5}$, the optimizer is Adam, and dropout = 0.2. For the FD002, the generator learning rate $\text{lr}_G = 1.2e^{-3}$, the discriminator learning rate $\text{lr}_D = 1.2e^{-1}$, and the rest of parameters are the same as FD001.

[6]https://scikit-learn.org/stable/modules/impute.html

E. Simulation Results

1) Data Reconstruction Experiments: Fig. 11 and Table III show the comparison of RMSE imputation effects between the proposed method and the baseline methods on the FD001, FD002, and FD003 datasets, where each method is trained–tested five times at each MR, and it can be seen that the PPCTE-TSGAIN model achieves the best imputation results at the vast majority of MRs on the three datasets.

Specifically, median imputation and mean imputation are the worst among the baseline methods, which may be due to the failure of the two methods to effectively mine the information in the dataset. KNN imputation [10] and MICE imputation [11] are both superior at low MR, but their imputation effect deteriorates rapidly with the increase in MR, which may be due to the increase of MR destroys the internal relationship of data required by the two methods. GAIN [17] has been proved to have good imputation results on many nonsequential datasets, but the imputation results on aeroengine sensor data are poor due to its inability to extract data timing information. Brits [14], GRU-TSGAIN, and PPCTE-TSGAIN are imputation models for time-series data. It can be seen that the imputation accuracy of these three models decreases slowly with the increase in the MR, which reflects the advantages of temporal models in aeroengine data imputation work.

In addition, PPCTE-TSGAIN achieved the best imputation results (the average RMSE indicator decreased by 27.26%, 84.7%, and 38.82%, respectively, compared with KNN imputation; 21.42%, 45.86%, and 27.97%, respectively, compared with MICE; 4.44%, 63.30%, and 9.11%, respectively, compared with GRU-TSGAIN; and 1.45%, 63.37%, and 5.32%, respectively, compared with Brits). It can be seen that the imputation accuracy of the proposed PPCTE-TSGAIN model on the FD001 and FD003 datasets is only slightly higher than that of the other models. This may be because the FD001 and FD003 datasets have fewer effective sensors, and the internal information of the data is relatively simple, so the performance of the three models is close. On the contrary, because FD002 has the most effective sensors and it is the most complex of the three datasets, the proposed PPCTE-TSGAIN model shows great advantages in this dataset.

2) Visualization of Imputation Results: In order to show our imputation work more intuitively, we restore the data to the original scale through: $(X_{\text{imp}}^{\text{origin}})_{ij} = (X_{\text{imp}})_{ij} \times \sigma_j + \mu_j$, where, for $\forall i \in \{1, 2, \ldots, Tw\}$, $j \in \{1, 2, \ldots, F\}$, $(X_{\text{imp}}^{\text{origin}})_{ij}$

TABLE III

RMSE Comparison Between the Proposed PPCTE-TSGAIN and the Baseline Algorithm on FD001, FD002, and FD003. The Underlined Value Is the Preimputation Algorithm Result. The Bold Value Is the Best Result. IMP Is the Average Improvement of the Proposed Model Over the Baseline Model

FD001\MR	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	average	IMP
Median	1.0071	1.0141	1.0003	1.0031	1.0084	1.0052	1.0050	1.0081	1.0062	1.0064	43.41%
Mean	1.0024	1.0079	0.9946	0.9981	1.0019	1.0004	0.9998	1.0008	1.0006	1.0007	43.09%
KNN [10]	0.5694	0.5918	0.6337	0.7010	0.7718	0.7912	0.7981	0.7971	0.8688	0.7248	21.42%
MICE [11]	0.5335	0.5509	0.5642	0.6275	0.6551	0.6787	0.7323	0.8914	0.9850	0.6910	17.58%
GAIN [17]	0.8252	0.8388	0.8376	0.8491	0.8583	0.8696	0.8871	0.9213	0.9941	0.8757	34.96%
Brits [14]	0.5560	0.5542	0.5565	0.5596	0.5623	0.5749	0.5855	0.5992	**0.6526**	0.5779	1.45%
GRU-TSGAIN	0.5684	0.5621	0.5698	0.5790	0.5899	0.5996	0.6013	0.6261	0.6675	0.5960	4.44%
PPCTE-TSGAIN	**0.5293**	**0.5344**	**0.5442**	**0.5499**	**0.5549**	**0.5650**	**0.5748**	**0.5961**	0.6770	**0.5695**	—
FD002\MR	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	average	IMP
Median	1.1068	1.1053	1.1031	1.1035	1.1046	1.1042	1.1027	1.1033	1.1032	1.1041	86.49%
Mean	1.0012	1.0019	1.0000	1.0000	1.0003	1.0008	1.0001	1.0003	1.0003	1.0005	85.09%
KNN [10]	0.0615	0.0639	0.0782	0.1727	0.3087	0.3976	0.4365	0.4476	**0.5126**	0.2755	45.86%
MICE [11]	0.0725	0.0757	0.0916	0.1153	0.1496	0.2008	0.2917	0.8067	0.9742	0.3087	51.68%
GAIN [17]	0.7585	0.7615	0.7605	0.7731	0.7772	0.7742	0.7819	0.8002	0.8725	0.7844	80.99%
Brits [14]	0.1914	0.2386	0.2770	0.3213	0.3673	0.4245	0.5014	0.6009	0.7418	0.4071	63.37%
GRU-TSGAIN	0.1729	0.1780	0.2040	0.2488	0.3207	0.4053	0.5478	0.7126	0.8676	0.4064	63.30%
PPCTE-TSGAIN	**0.0612**	**0.0601**	**0.0626**	**0.0647**	**0.0755**	**0.0914**	**0.1401**	**0.2706**	0.5160	**0.1491**	—
FD003\MR	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	average	IMP
Median	1.0210	1.0213	1.0237	1.0160	1.0223	1.0166	1.0201	1.0250	1.0209	1.0208	51.39%
Mean	0.9974	1.0005	1.0019	0.9996	1.0001	0.9991	1.0001	1.0014	1.0019	1.0002	50.39%
KNN [10]	0.4494	0.4661	0.5028	0.6150	0.7708	0.8318	0.8529	0.8519	0.8586	0.6888	27.97%
MICE [11]	0.4701	0.4854	0.5146	0.5836	0.6132	0.6526	0.7273	0.9263	0.9712	0.6605	24.88%
GAIN [17]	0.8023	0.8151	0.8162	0.8426	0.8402	0.8482	0.8817	0.9132	0.9896	0.8610	42.37%
Brits [14]	0.4767	0.4818	0.4817	0.4999	0.5058	0.5142	0.5404	0.5760	0.6401	0.5241	5.32%
GRU-TSGAIN	0.5012	0.5052	0.5101	0.5297	0.5291	0.5467	0.5645	0.5876	0.6390	0.5459	9.11%
PPCTE-TSGAIN	**0.4327**	**0.4411**	**0.4540**	**0.4607**	**0.4757**	**0.4878**	**0.5232**	**0.5691**	**0.6211**	**0.4962**	—

is the ith row and the jth column elements of the original scale imputed data $X_{\text{imp}}^{\text{origin}} \in \mathbb{R}^{Tw \times F}$, and then, we splice each piece of $X_{\text{imp}}^{\text{origin}}$ to $\mathbb{R}^{N \times F}$, and finally, we compare it with the original data $X^{\text{raw}} \in \mathbb{R}^{N \times F}$.

Fig. 12 shows the comparison of the original values, the missing values, the KNN imputation values, and the PPCTE-TSGAIN imputation values under the condition of 50% MR, where the engines are randomly selected in the test set. Specifically, Fig. 12 shows the data of No. 200 engine in FD001, the data of No. 228 engine in FD002, and the data of No. 50 engine in FD003.

For the imputation work of No. 200 engine in FD001 and No. 51 engine in FD003, it can be seen that the distribution of the data imputed by KNN imputation is very different from the original data, which may be due to the serious reduction of the effective information at 50% MR, resulting in the destruction of the associated information required by KNN imputation. On the contrary, the PPCTE-TSGAIN imputed data are not only closer to the missing data, but also more consistent with the distribution trend of the original data. For the imputation work of No. 228 engine in FD002, KNN imputation results showed many values outside the original data distribution, while the PPCTE-TSGAIN imputation results are highly coincident with the distribution of the original data and have high accuracy. In general, the above results directly prove the superiority of our model in the imputation of complex MTS data, such as aeroengine data.

3) Downstream Task Evaluation Experiments: In order to fully evaluate the effectiveness of our missing data imputation work and the impact of imputation results on downstream tasks, the imputed data are used to predict the RUL of aeroengines, and the experiment is shown in Fig. 13.

First, we train three fixed parameter RUL prediction models for FD001, FD002, and FD003 using the method proposed in [30] (i.e., using transformer as a predictor) according to the partitioning rules of the C-MAPSS test set and training set shown in Table IV.

Second, we simulate three conditions of 0.1, 0.3, and 0.5 MRs and impute missing test set with different MR conditions, obtaining the RMSE accuracy of data reconstruction and the imputed test set (compared with the imputation RMSE in data reconstruction experiment, there is a certain change in the imputation RMSE in this experiment; this is because we have reclassified the dataset according to the requirements of the RUL prediction task).

Finally, we input the complete test set and the imputed test set into the corresponding RUL prediction model to obtain the RUL prediction results and use the RMSE and score indicators of the RUL to evaluate the quality of the reconstructed test set. The RMSE and score indicators for the RUL are defined as follows:

$$\text{RUL RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{\text{n}} d_i^2} \tag{30}$$

$$\text{RUL Score} = \begin{cases} \sum_{i=1}^{n}\left(e^{-\frac{d_i}{13}} - 1\right), & d_i < 0 \\ \sum_{i=1}^{n}\left(e^{\frac{d_i}{10}} - 1\right), & d_i \geq 0 \end{cases} \tag{31}$$

where n is the number of engine samples in the test set and d_i is the difference between the predicted value and the actual value of the RUL values. RUL RMSE is the general evaluation indicator for regression tasks, and RUL score is the practical

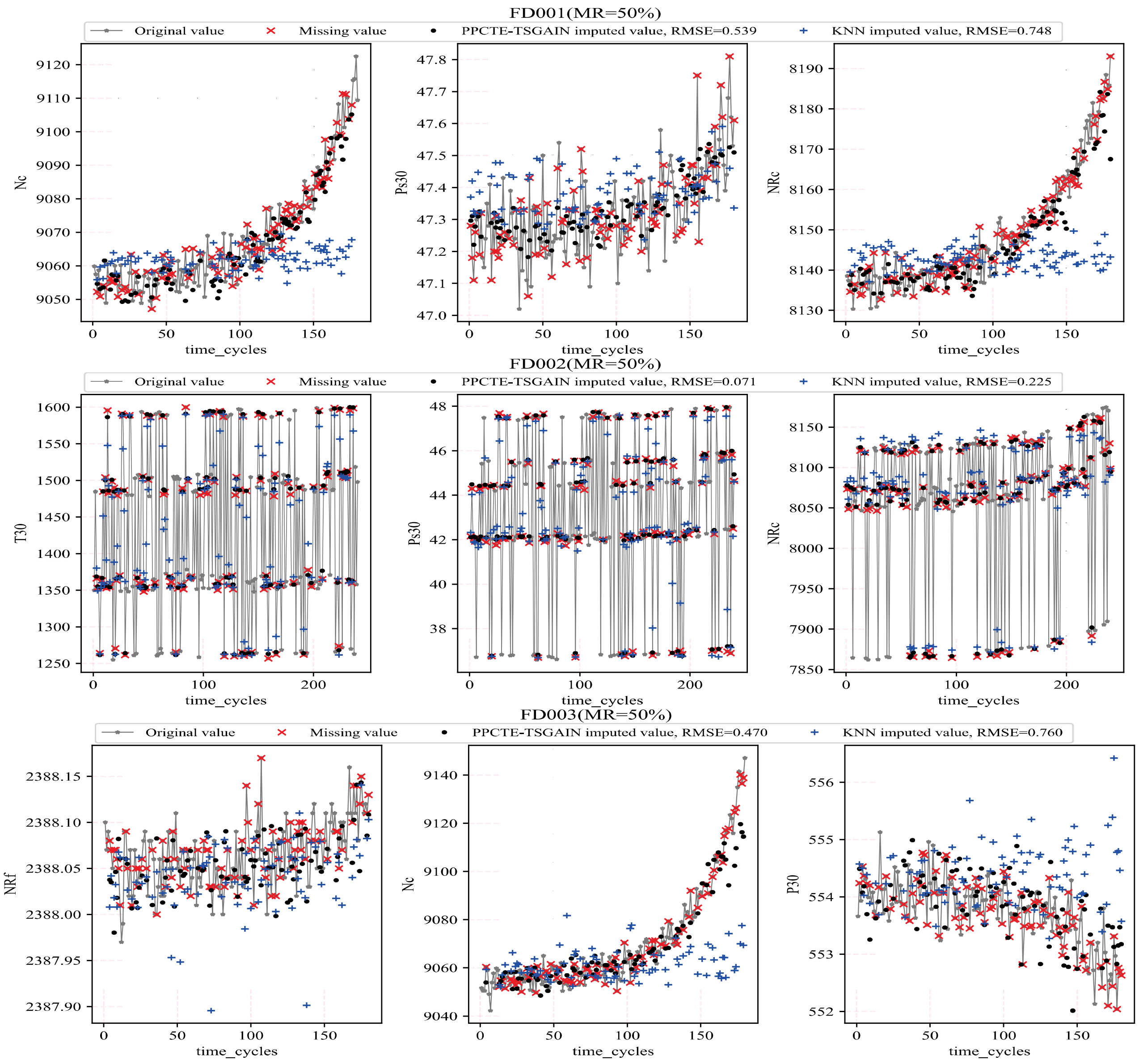


Fig. 12. Imputation results of No. 200 engine in FD001, No. 228 engine in FD002, and No. 51 engine in FD003 at 50% MR (three sensors are randomly displayed, respectively).

TABLE IV

DATASET PARTITIONING DURING DOWNSTREAM TASK EVALUATION

Subsets	FD001	FD002	FD003
Number of training engines	100	260	100
Number of testing engines	100	259	100

application evaluation indicator (because overestimating RUL has a greater impact than underestimating RUL). Specially, the smaller their values, the more accurate the model predictions are. When the prediction models are the same, it can indirectly indicate that the quality of input data is higher.

Table V shows the RUL prediction accuracy of the complete test set data for FD001, FDD002, and FD003, which will be the benchmark data for our experiment.

In order to study the impact of different imputation methods on downstream tasks, we input the test set imputed by different imputation methods into a fixed model for RUL prediction.

TABLE V

RMSE AND SCORE PERFORMANCE OF FD001, FD002, AND FD003 RUL PREDICTION MODELS ON THE TEST DATASET WITHOUT MISSING VALUES

Dataset \ Indicators	RUL RMSE	RUL Score
FD001	16.59	558.64
FD002	19.479	1334.7
FD003	17.67	917.03

Table VI shows the accuracy of using different imputation methods to reconstruct the test set for FD002 at a 50% MR, as well as the accuracy of predicting RUL using imputed test set.

From Table VI, we can see that, generally, the higher the imputation accuracy (i.e., the smaller the imputation RMSE), the higher the accuracy of RUL prediction (i.e., the smaller the RUL RMSE and RUL score), and the better the quality of the imputation data. It is worth noting

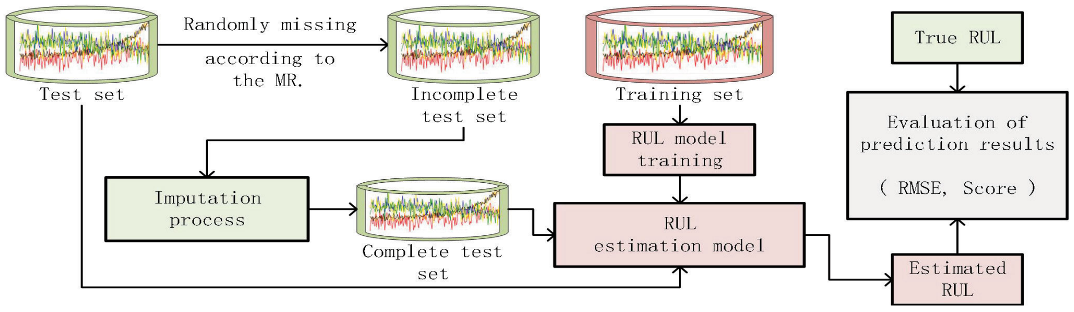


Fig. 13. Flowchart for predicting the RUL of incomplete data.

TABLE VI

UNDER THE MR OF 0.5, THE COMPARISON OF IMPUTATION ACCURACY OF DIFFERENT METHODS FOR THE FD002 DATASET AND THE COMPARISON OF PREDICTION ACCURACY OF RUL OF DIFFERENT IMPUTED DATA. THE LAST ROW IS THE CONTROL GROUP, I.E., THE ACCURACY OF PREDICTING THE RUL OF THE COMPLETE ORIGINAL TEST SET

Methods \ Indicators	Imputation RMSE	RUL RMSE	RUL Score
Median	1.104	75.128	530432.6
Mean	0.999	74.245	486506.0
KNN [10]	0.237	42.254	39506.7
Mice [11]	0.162	27.372	4349.8
GAIN [17]	0.758	68.301	325424.4
Brits [14]	0.367	48.902	59977.2
GRU-TSGAIN	0.312	39.361	18785.7
PPCTE-TSGAIN	**0.081**	**21.787**	**2158.7**
Origin data	0	19.479	1334.7

that at a 50% MR, the predicted results of the imputation data of the proposed PPCTE-TSGAIN are RUL RMSE/21.787 and RUL score/2158.7, which is very close to the RUL RMSE/19.479 and RUL score/1334.7 using the complete test set, and far better than other imputation methods. This proves that the proposed PPCTE-TSGAIN not only has higher imputation accuracy but also is suitable for downstream tasks.

To verify the adaptability of PPCTE-TSGAIN to downstream tasks under different MRs, we set three MR conditions of 0.1, 0.3, and 0.5, and conducted simulation experiments on three datasets: FD001, FD002, and FD003. Table VII shows the performance results of the proposed PPCTE-TSGAIN method under different MRs.

From Table VII, it can be seen that as the MR increases, the imputation RMSE gradually increases, and the overall prediction accuracy of the remaining service life also decreases, which is consistent with our expectations. In addition, we have two new findings.

1) Under certain MR conditions, the prediction accuracy of the imputed dataset is even better than that of the original dataset, for example, the RUL RMSE indicator: [FD002, MR 0.1 versus nonmissing (19.068 versus 19.479)] and [FD003, MR 0.1 versus nonmissing (17.253 versus 17.670)]. RUL score indicators: [FD001, MR 0.1 versus nonmissing (532.1 versus 558.64)] and [FD002, MR 0.1 versus nonmissing (1241.1 versus 1334.7)]. This may be due to one of the optimization goals of our imputation method being MSE loss, which enables the imputed data (as shown in Fig. 12) to have fewer outliers, resulting in prediction accuracy that may even exceed the original data when the MR is low.
2) Although the imputation RMSE values of FD001 and FD003 are much higher than those of FD002, they have achieved performance close to FD002 in downstream tasks. This is mainly caused by the distribution of the data itself. From Fig. 12, it can be seen that the data of FD001 and FD003 are relatively scattered. Therefore, although the reconstructed values falls within the overall distribution, there may be a certain gap with the true values, resulting in a larger imputation RMSE. The FD002 data distribution is relatively concentrated (with multiple bar distributions), so the imputation RMSE is relatively small. But, imputation RMSE only measures the quality of imputation results from one aspect, and the inherent nature of the dataset can lead to imputation RMSE having a different range. From the perspective of downstream tasks, the imputation results of FD001 and FD003 still perform well.

These findings demonstrate that the proposed PPCTE-TSGAIN not only has excellent imputation accuracy, but also has good adaptability to downstream tasks.

F. Discussions

In this section, we will conduct comparative experiments and ablation experiments to analyze the role of each designed module.

1) TE Versus GRU: In order to fairly compare the TE [27] module with the GRU [28] (a kind of traditional recurrent neural network) module, we use the TE module and the GRU module as the generator and discriminator of the TSGAIN framework, called TE-TSGAIN and GRU-TSGAIN, respectively. Then, for the three subsets, we independently simulate five times under each MR and took the average value, and finally compare the imputation effects of the TE-TSGAIN model and the GRU-TSGAIN model, as shown in Fig. 14.

It can be seen from Fig. 14 that with the increase in the MR, the overall performance of the two temporal modules gradually decreases. However, their specific performance is different. Specifically, the performance of TE-TSGAIN model in FD001 and FD002 datasets is better than that of the GRU-TSGAN model, while in FD003, the TE-TSGAIN model shows similar performance to that of the GRU-TSGA model. It is worth noting that although the GRU-TSGAIN model can

TABLE VII

RUL Prediction of FD001, FD002, and FD003 Test Sets Was Conducted Under the Conditions of No Missing, MR 0.1, MR 0.3, and MR 0.5, Respectively. The Imputation RMSE Is the Data Reconstruction Accuracy, and the RUL RMSE and RUL Score Are the Remaining Service Life Accuracy

Indicators \ Dataset	FD001				FD002				FD003			
Missing rate	0	0.1	0.3	0.5	0	0.1	0.3	0.5	0	0.1	0.3	0.5
Imputation RMSE	0	0.645	0.649	0.656	0	0.059	0.076	0.081	0	0.533	0.545	0.556
RUL RMSE	16. 59	17. 435	19. 536	24.14	19. 479	19. 068	19. 954	21. 787	17.67	17.253	18. 109	18. 160
RUL Score	558.64	532.1	805.3	1591.2	1334.7	1241. 1	1364.9	2158.7	917.03	993. 0	1016. 7	1106. 4

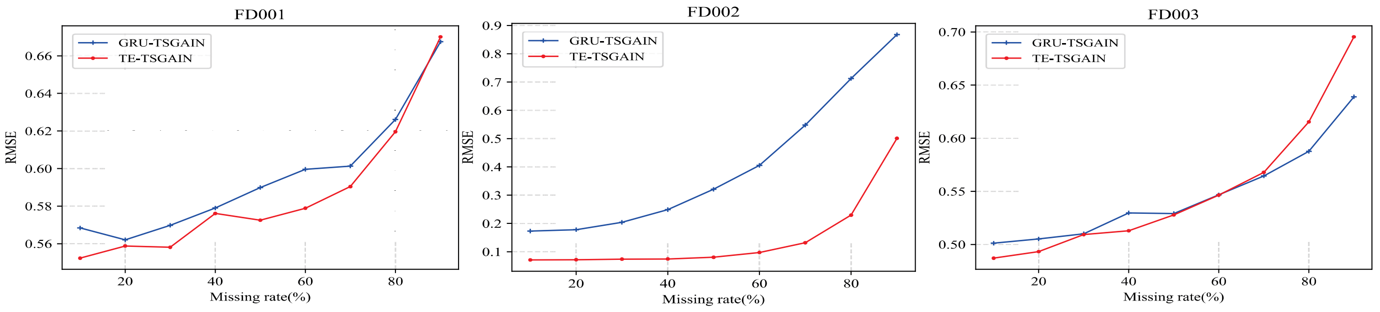


Fig. 14. RMSE comparison results of GRU-TSGAIN and TE-TSGAIN on FD001, FDS002, and FD003 datasets.

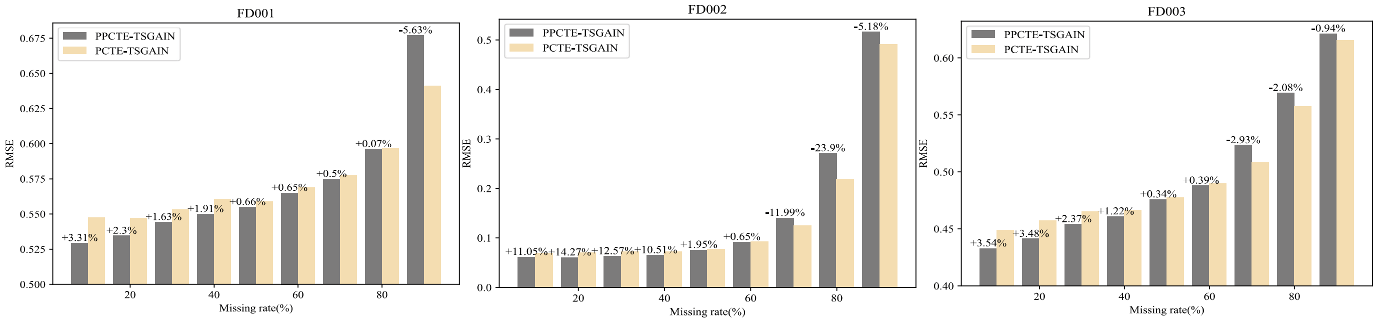


Fig. 15. Experimental results of the preimputation module ablation.

produce certain results on relatively simple datasets (such as FD001 and FD003), in FD002, the GRU-TSGIN model is far worse than the TE-TSGAIN model. This may be because the number of sensors in FD002 is larger, the data association is more complex, and the information extraction ability of the GRU module is insufficient. The above findings highlight the effectiveness of the TE module with attention mechanism in mining missing data imputation information of aeroengines, so as to position it as a more appropriate time extraction module.

2) Ablation of the Preimputation Module: To evaluate the contribution of the preimputation module, we delete the preimputation module of the PPCTE-TSGAIN model and call it PCTE-TSGAIN. Then, we independently simulate five times under each MR condition and take the average value, and all overlapping parameters of the PCTE-TSGAIN model and the PPCTE-TSGAIN model are the same. Finally, the imputation results of the PCTE-TSGAIN model and the PPCTE-TSGAIN model are compared, as shown in Fig. 15, where the label value of the bar graph shows the improvement degree of PCTE-TSGAIN brought by the preimputation module.

It can be seen that under specific MR conditions (i.e., FD001 MR $<$ 90%, FD002 MR $<$ 70%, and FD003 MR $<$ 70%), the preimputation module can effectively improve the imputation accuracy of the proposed model. This can be attributed to the improved accuracy of input information provided by the preimputation module during model training, allowing the proposed model to leverage the strengths of the preimputation module. However, under the condition of high MR, the imputation accuracy of the proposed model is inevitably affected due to the significant reduction of the imputation accuracy of the preimputation module.

However, this effect can be ignored to a certain extent for the following reasons: 1) the degradation of accuracy always occurs at high MRs, and in real life, it is relatively rare for data to have such high MRs; 2) when the data reach a certain MR, researchers will discard it, because it contains too little information; and 3) the proposed model still can achieve better imputation accuracy at high MRs compared with other baseline imputation methods. Therefore, we focus more attention on the improvement of the results at lower MRs.

3) Ablation of the PC Module: In order to evaluate the contribution of the PC module, we remove the PC module from the PCTE-TSGAIN model (i.e., TE-TSGAIN). Then, we independently simulate five times under each MR condition and take the average value, where all overlapping parameters of the PCTE-TSGAIN model and the TE-TSGAIN model are the same. Fig. 16 shows the comparison of results, where the

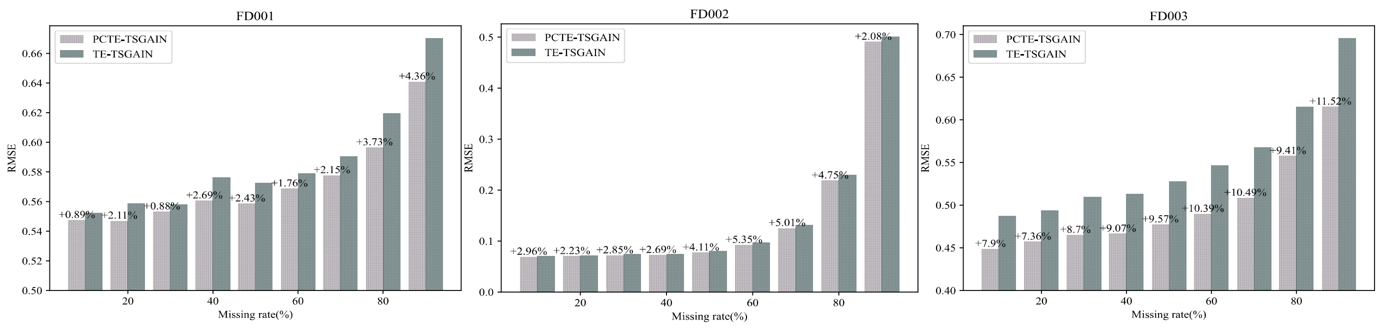


Fig. 16. Experimental results of the PC module ablation.

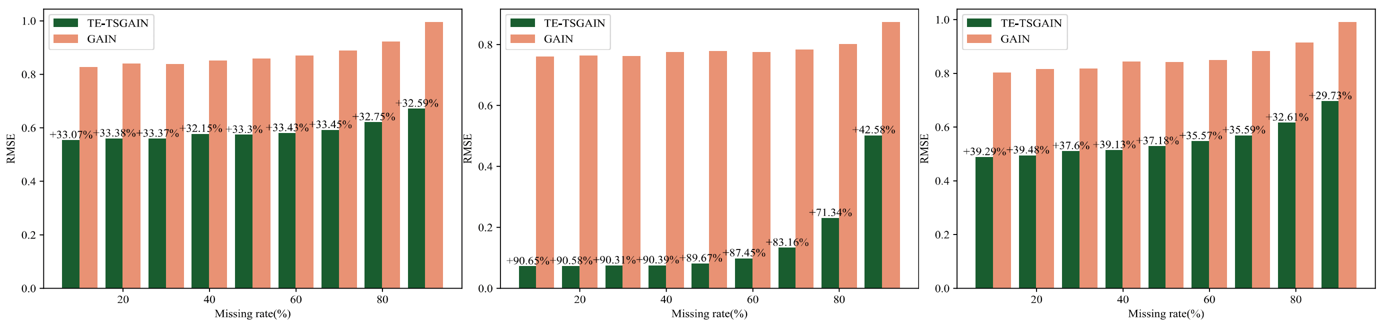


Fig. 17. RMSE comparison results of TE-TSGAIN and GAIN on FD001, FDS002, and FD003 datasets.

label value of the bar graph shows the improvement degree of TE-TSGAIN brought by the PC module.

The results demonstrate that compared with the TE-TSGAIN model, the imputation accuracy of PCTE-TSGAIN model has a certain improvement under all the MRs of the three datasets. Furthermore, Fig. 16 reveals that the improvement effect of the PC module will not weaken or disappear with the increase in the MR, which proves that the learning effect of PC on feature correlation information is obvious. Therefore, strengthening the extraction of different feature correlation information will help to improve the imputation accuracy and ultimately provide more accurate information for subsequent data analysis.

4) Comparison of the TSGAIN and GAIN: To assess the advantage of TSGAIN's spatiotemporal structure over GAIN, we choose TE-TSGAIN and GAIN for contrastive experiments. Specifically, we repeat the simulation five times under different loss rate conditions and take the average value. Fig. 17 presents the comparison of results, where the tag value of the bar chart indicates the improvement percentage of TE-TSGAIN over GAIN.

The results show that on FD001, FD002, and FD003 with spatiotemporal features, TE-TSGAIN outperforms GAIN significantly. This is attributed to the TSGAIN framework's full preservation and utilization of time information, and the TE module's effective extraction of time information. It should be noted that GAIN's generator and discriminator consist of MLP modules, while TE-TSGAIN's generator and discriminator consist of TE modules. This is because GAIN is a framework designed for nonsequential data, while TSGAIN is a framework designed for sequential data, making it hard to unify the structure of generators and discriminators. However, from the results, it can be seen that TSGAIN extends the applicability of GAIN to datasets with spatiotemporal features, enabling it to fully exploit the time information of the data in missing data imputation work.

V. Conclusion

In this article, we have proposed a novel end-to-end imputation model called PPCTE-TSGAIN for the imputation of missing data. The evaluations on the C-MAPSS dataset demonstrate that the PPCTE-TSGAIN model has high accuracy in the missing data imputation of MTS, and the visualization experiment proves that the data imputed by PPCTE-TSGAIN model have a strong similarity with the distribution of the original data. In addition, the developed TSGAIN framework, preimputation module, PC module, and TE module have been proved to be advantageous for the information extraction and imputation of missing data. These findings highlight the effectiveness of PPCTE-TSGAIN in imputing missing values of MTS, which can ultimately provide more accurate information for the field of industrial artificial intelligence.

References

[1] T. Sun and X.-M. Sun, "New results on classification modeling of noisy tensor datasets: A fuzzy support tensor machine dual model," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 8, pp. 5188–5200, Aug. 2022.

[2] T. Sun, X.-M. Sun, and A. Sun, "Optimal output tracking of aircraft engine systems: A data-driven adaptive performance seeking control," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 3, pp. 1467–1471, Mar. 2022.

[3] T. Wang, D. Guo, and X.-M. Sun, "Remaining useful life predictions for turbofan engine degradation based on concurrent semi-supervised model," *Neural Comput. Appl.*, vol. 34, no. 7, pp. 5151–5160, Apr. 2022.

[4] C. Chen, N. Lu, B. Jiang, Y. Xing, and Z. H. Zhu, "Prediction interval estimation of aeroengine remaining useful life based on bidirectional long short-term memory network," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, 2021.

[5] L. Ren, H. Qin, Z. Xie, B. Li, and K. Xu, "Aero-engine remaining useful life estimation based on multi-head networks," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, 2022.

[6] S. Zhang, R. Kang, X. He, and M. G. Pecht, "China's efforts in prognostics and health management," *IEEE Trans. Compon. Packag. Technol.*, vol. 31, no. 2, pp. 509–518, Jun. 2008.

[7] A. V. Rao et al., "Challenges in engine health monitoring instrumentation during developmental testing of gas turbine engines," in *Proc. Nat. Aerosp. Propuls. Conf.* Cham, Switzerland: Springer, 2021, pp. 275–295.

[8] Y. Huang, Y. Tang, J. VanZwieten, and J. Liu, "Reliable machine prognostic health management in the presence of missing data," *Concurrency Comput., Pract. Exper.*, vol. 34, no. 12, p. e5762, May 2022.

[9] J. W. Graham, "Missing data analysis: Making it work in the real world," *Annu. Rev. Psychol.*, vol. 60, no. 1, pp. 549–576, Jan. 2009.

[10] A. T. Hudak, N. L. Crookston, J. S. Evans, D. E. Hall, and M. J. Falkowski, "Nearest neighbor imputation of species-level, plot-scale forest structure attributes from LiDAR data," *Remote Sens. Environ.*, vol. 112, no. 5, pp. 2232–2245, May 2008.

[11] S. V. Buuren and K. Groothuis-Oudshoorn, "Mice: Multivariate imputation by chained equations in R," *J. Stat. Softw.*, vol. 45, no. 3, pp. 1–67, 2011.

[12] P. K. Sharpe and R. J. Solly, "Dealing with missing values in neural network-based diagnostic systems," *Neural Comput. Appl.*, vol. 3, no. 2, pp. 73–77, Jun. 1995.

[13] C. Gautam and V. Ravi, "Data imputation via evolutionary computation, clustering and a neural network," *Neurocomputing*, vol. 156, pp. 134–142, May 2015.

[14] 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. 6776–6786.

[15] Y. Zhang, B. Zhou, X. Cai, W. Guo, X. Ding, and X. Yuan, "Missing value imputation in multivariate time series with end-to-end generative adversarial networks," *Inf. Sci.*, vol. 551, pp. 67–82, Apr. 2021.

[16] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Sci. Rep.*, vol. 8, no. 1, pp. 1–12, Apr. 2018.

[17] J. Yoon, J. Jordon, and M. van der Schaar, "GAIN: Missing data imputation using generative adversarial nets," 2018, *arXiv:1806.02920*.

[18] Y. Wang, D. Li, X. Li, and M. Yang, "PC-GAIN: Pseudo-label conditional generative adversarial imputation networks for incomplete data," *Neural Netw.*, vol. 141, pp. 395–403, Sep. 2021.

[19] W. Du, D. Côté, and Y. Liu, "SAITS: Self-attention-based imputation for time series," *Exp. Syst. Appl.*, vol. 219, Jun. 2023, Art. no. 119619.

[20] A. Y. Yildiz, E. Koç, and A. Koç, "Multivariate time series imputation with transformers," *IEEE Signal Process. Lett.*, vol. 29, pp. 2517–2521, 2022.

[21] Y. Shi, J. Zhao, and X.-M. Sun, "A bumpless transfer control strategy for switched systems and its application to an aero-engine," *IEEE Trans. Ind. Informat.*, vol. 17, no. 1, pp. 52–62, Jan. 2021.

[22] M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf, "Multiple imputation by chained equations: What is it and how does it work?" *Int. J. Methods Psychiatric Res.*, vol. 20, no. 1, pp. 40–49, Mar. 2011.

[23] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[24] L. Tunstall, L. Von Werra, and T. Wolf, *Natural Language Processing With Transformers*. Sebastopol, CA, USA: O'Reilly Media, 2022.

[25] H. Shang, J. Wu, C. Sun, J. Liu, X. Chen, and R. Yan, "Global prior transformer network in intelligent borescope inspection for surface damage detection of aeroengine blade," *IEEE Trans. Ind. Informat.*, vol. 19, no. 8, pp. 8865–8877, Aug. 2023.

[26] Z. Lian, B. Liu, and J. Tao, "CTNet: Conversational transformer network for emotion recognition," *IEEE/ACM Trans. Audio, Speech, Lang., Process.*, vol. 29, pp. 985–1000, 2021.

[27] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–11.

[28] Y. Luo et al., "Multivariate time series imputation with generative adversarial networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1596–1607.

[29] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 10, pp. 2825–2830, Jul. 2017.

[30] L. Liu, X. Song, and Z. Zhou, "Aircraft engine remaining useful life estimation via a double attention-based data-driven architecture," *Rel. Eng. Syst. Saf.*, vol. 221, May 2022, Art. no. 108330.

Song Ma received the Ph.D. degree from the National University of Defense Technology, Changsha, China, in 2019.

He is currently a Researcher with the Dalian University of Technology, Dalian, China. He has presided over the key projects of the national key research and development plan and many other national, provincial, and ministerial projects. He has already published more than 30 academic papers. His research interests include aeroengine health management, aviation artificial intelligence, aviation composite diagnosis based on big data, and so on.

Zeng-Song Xu received the B.S. degree in control theory and control engineering from the School of Control Science and Engineering, Dalian University of Technology, Dalian, China, in 2021, where he is currently pursuing the M.S. degree in control theory and control engineering.

His research interests include industrial artificial intelligence, deep learning, missing time-series data imputation and reconstruction, aeroengine data analysis, and so on.

Tao Sun received the M.S. degree in operational research and cybernetics from the School of Mathematical Sciences, Dalian University of Technology, Dalian, China, in 2017, and the Ph.D. degree in control theory and control engineering from the School of Control Science and Engineering, Dalian University of Technology, in 2022.

He is currently a Post-Doctoral Researcher with the Department of Automation, Tsinghua University, Beijing, China. His research interests include brain-inspired intelligence, complex systems, aeroengine control and diagnosis, and so on.

Dr. Sun serves as a reviewer for *Mathematical Reviews* in the American Mathematical Society and an Editorial Board Member for the *Journal of Modern Industry and Manufacturing*, the *International Journal of Systems Engineering*, the *International Journal of Data Science and Analysis*, and *Machine Learning Research*.