Imputation of Missing Values in Time Series Using an Adaptive-Learned Median-Filled Deep Autoencoder

Zhuofu Pan, Yalin Wang, *Member, IEEE*, Kai Wang, *Member, IEEE*, Hongtian Chen, *Member, IEEE*, Chunhua Yang, *Fellow, IEEE*, and Weihua Gui

***Abstract*—Missing values are ubiquitous in industrial data sets because of multisampling rates, sensor faults, and transmission failures. The incomplete data obstruct the effective use of data and degrade the performance of data-driven models. Numerous imputation algorithms have been proposed to deal with missing values, primarily based on supervised learning, that is, imputing the missing values by constructing a prediction model with the remaining complete data. They have limited performance when the amount of incomplete data is overwhelming. Moreover, many methods have not considered the autocorrelation of time-series data. Thus, an adaptive-learned median-filled deep autoencoder (AM-DAE) is proposed in this study, aiming to impute missing values of industrial time-series data in an unsupervised manner. It continuously replaces the missing values by the median of the input data and its reconstruction, which allows the imputation information to be transmitted with the training process. In addition, an adaptive learning strategy is adopted to guide the AM-DAE paying more attention to the reconstruction learning of nonmissing values or missing values in different iteration periods. Finally, two industrial examples are used to verify the superior performance of the proposed method compared with other advanced techniques.**

***Index Terms*—Adaptive-learned median-filled deep autoencoder (AM-DAE), industrial-type missing values, time-series imputation, unsupervised imputation.**

Manuscript received 13 January 2022; accepted 13 April 2022. Date of publication 4 May 2022; date of current version 13 January 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1713800; in part by the National Natural Science Foundation of China under Grant U1911401 and Grant 61590921; in part by the Science and Technology Innovation Program of Hunan Province in China under Grant 2021RC4054; and in part by the China Scholarship Council under Grant 202006370200. This article was recommended by Associate Editor H. Han. *(Corresponding authors: Yalin Wang; Kai Wang.)*

Zhuofu Pan is with the School of Automation, Central South University, Changsha 410083, China, and also with the Department of Chemical and Materials Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada (e-mail: panfuzz@csu.edu.cn).

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

Hongtian Chen is with the Department of Chemical and Materials Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada (e-mail: chtbaylor@163.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2022.3167995.

Digital Object Identifier 10.1109/TCYB.2022.3167995

I. Introduction

With the popularization of distributed control systems, industrial production has become more automated and intelligent. A large amount of accessible data inspires the widespread use of data-driven models in industrial process optimization, prediction, diagnosis, and control [1]–[4]. However, due to multiple sampling rates and unexpected failures in sensors or transmissions, the data collected from industrial processes often contain missing values. These missing items obstruct the completeness of data and cause the model to learn an inaccurate distribution [5], [6].

In general, there are a variety of approaches to dealing with missing values. The most straightforward approach is to delete the variables that contain missing items. This strategy is convenient for modeling, but it risks erroneously deleting crucial variables, resulting in a less informative model. In contrast, imputation, which imputes missing values by learning data characteristics from nonmissing values, can completely use all available values. Typically, imputation methods are classified as statistical analysis-based, machine learning-based, and deep learning-based methods [7], [8].

Many statistical approaches, such as mean imputation, regression imputation, multiple imputations, etc., were used to impute missing values. They are rough estimates of individual variables, ignoring the spatiotemporal correlations between variables. Hence, they are commonly utilized as initial values of missing values in machine learning- and deep learning-based imputation methods. K-nearest neighbor [9], self-organizing map [10], and support vector machine [11] are some representatives of machine learning-based imputation methods. They are all based on supervised imputation learning, that is, the model is trained with complete data and tested with incomplete data, in which the nonmissing values in the incomplete samples are excluded from the training process. The potentially valuable information in the nonmissing values is wasted, resulting in a degraded imputation performance. Of course, the generalizability of the model and the ability to extract complex features are also crucial factors affecting imputation performance. Machine learning-based approaches are limited in extracting valuable features, especially with the multivariate time-series data collected from complicated systems [12]–[14]. In contrast, deep learning has greater expressiveness, leading to a more popular data-driven model.

Deep learning-based approaches have also been developed for imputations [7], [15]. For example, Ma *et al.* [16] investigated the correlations between missing and nonmissing values using deep denoising autoencoders. Also, Yoon *et al.* [17] developed a generative adversarial imputation network (GAIN) in which the discriminator is designed to discriminate whether the value is real or imputed. Then, Fortuin *et al.* [18] utilized the deep variational autoencoder and Gaussian process to transform the incomplete time-series data into a low-dimensional space for better imputation. In addition, Spinelli *et al.* [19] constructed the similarities of graphs between the complete and incomplete data set using the graph imputation neural network. These methods demonstrate excellent expression capabilities of deep learning-based methods. They can impute well for incomplete data with complicated nonlinearity and time-series characteristics. However, most of them employ a supervised training strategy, in which the complete data are randomly missed as incomplete data to feed into the model, and the original complete data are used as supervised information to correct the reconstruction. It is certainly not the case in reality. Moreover, when the amount of incomplete data is the majority, the training effectiveness of such strategy will plummet.

Aside from employing different models, another important factor in distinguishing different imputation performances is the update strategy of missing values. Different imputation strategies will result in varying imputation results even when utilizing the same model. So far, many imputation strategies have been proposed. For example, Lai *et al.* [20] developed a multitask learning mechanism that the output layer of the autoencoder not only performs classification tasks but also imputes missing values in the incomplete input. Also, Lai *et al.* [21] built a tracking-removed autoencoder (TRAE) to impute missing values with their gradients. The missing values can be updated like network parameters by backpropagating the error to the input layer. In addition, recursive updating of missing values is a widely adopted strategy, such as iterative generative adversarial networks for imputation [22], viewing imputation with generative adversarial networks [23], and TRAE [21]. Instead of imputing missing values by a single prediction, recursive updating allows continuously learning missing values during the training process. Although many deep learning-based imputation models and strategies have been proposed, superior imputation strategies for unsupervised learning are rare. Few studies have considered the time-series correlation of process data. Moreover, imputed values are not transmittable in most recursive learning, resulting in a lack of continuity and efficiency.

In order to address the above problems, an adaptive-learned median-filled deep autoencoder (AM-DAE) is presented to impute time-series missing values in this article. It provides a new unsupervised data imputation strategy based on DAE. Initially, the mean of the nonmissing values in each variable is used to initialize corresponding missing values. Then, the missing values are updated with the median of the input and its reconstruction after each forward propagation. In this way, previous imputed values can be transmitted with the recursive updating process, which improves the imputation efficiency and accuracy of AM-DAE. In addition, an adaptive loss function is adopted for parameter training in AM-DAE based on the statement that the model is hardly applicable to impute missing values until it learns the data distribution well. The adaptive loss function in AM-DAE allows it to pay more attention to reducing reconstruction errors of nonmissing values in the early training periods and eventually achieve a tradeoff between reconstruction and imputation learning. Also, AM-DAE is designed to match the characteristics of the time-series data. Stacked dynamic samples are employed to improve the imputation frequency of a single sample as well as the learning efficiency. Finally, two typical missing patterns in industrial data, that is, missing in fixed intervals and continuous periods, are simulated. The proposed approach is verified effectiveness in the Tennessee Eastman (TE) process and an actual hydrocracking process. The four-fold contributions of this article are summarized as follows.

1) Recursively impute missing values with the median of the input and reconstruction, allowing the imputation information to be transferred in the training process.
2) Use an adaptive loss function to focus on nonmissing value reconstruction or missing value imputation at different training periods.
3) Stack dynamic samples to increase the imputation frequency, thus improving the training efficiency and imputation performance.
4) Simulate imputations of industrial-type missing values. The effectiveness of the proposed method is verified.

The remainder of this article is organized as follows. Section II briefly reviews the deep autoencoder (DAE) approach for imputing missing values based on supervised learning. In Section III, the proposed approach is introduced. Also, the extension of AM-DAE to handle multivariate time-series missing values is described in this section. Section IV verifies the effectiveness and superior performance of AM-DAE compared to other advanced imputation methods using two imputation examples in chemical processes. Finally, Section V summarizes the primary contributions of this article.

II. Supervised Deep Autoencoder for Imputation

A. Deep Autoencoder

DAE is a multilayer neural network consisting of an encoder and a decoder. The encoder and decoder in DAE mine the hierarchical expression of input data and reconstruct the original data from compressed features, respectively. Unlike the encoder and decoder in autoencoder with a two-layer structure, the encoder and decoder in DAE commonly contain multiple layers. Fig. 1 depicts the overall structure of a basic DAE.

It can be seen that a DAE consists of an input layer, an output layer, and several hidden layers. Their activations can be denoted as $\boldsymbol{x}$, $\widetilde{\boldsymbol{x}}$, and $\{\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \ldots, \boldsymbol{h}^{(L)}, \boldsymbol{h}^{(L+1)}, \ldots, \boldsymbol{h}^{(o-1)}\}$, respectively. The blue arrows in Fig. 1 illustrate the encoding process, which aims to characterize the features with few dimensions from the original data as much as possible. The deep compressed features include the majority of information of the input data. Thanks to the multilayer nonlinear structure, the encoder can extract complicated features. Its forward

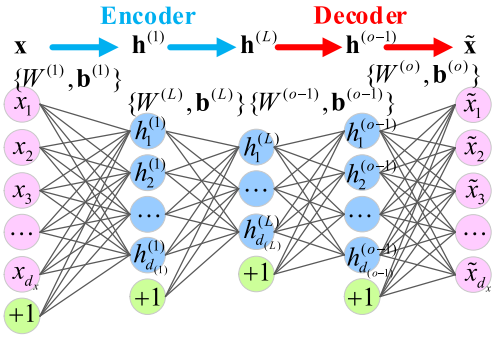


Fig. 1. Basic network structure of a DAE.

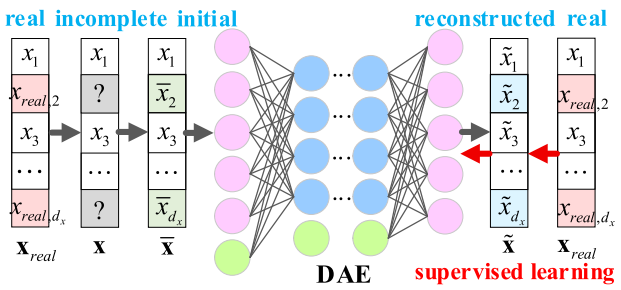


Fig. 2. DAE for imputation based on supervised learning.

propagation can be described by

$$\begin{aligned}\mathbf{h}^{(1)} &= \sigma^{(1)}\left(W^{(1)}\mathbf{x}+\mathbf{b}^{(1)}\right)\\ \mathbf{h}^{(i)} &= \sigma^{(i)}\left(W^{(i)}\mathbf{h}^{(i-1)}+\mathbf{b}^{(i)}\right), i=2,3,\ldots,L\end{aligned} \tag{1}$$

where $\boldsymbol{x} \in \boldsymbol{R}^{d_x}$ and $\boldsymbol{h}^{(i)} \in \boldsymbol{R}^{d_{(i)}}$ represent the input and the activation of the ith hidden layer in the encoder, respectively; d_x and $d_{(i)}$ denote their neuron numbers; $W^{(i)} \in \boldsymbol{R}^{d_{(i)}\times d_{(i-1)}}$ is a weight matrix connecting the $(i-1)$th and the ith layer; $\boldsymbol{b}^{(i)} \in \boldsymbol{R}^{d_{(i)}}$ stands for the bias of the ith hidden layer; $\sigma^{(i)}(\cdot)$ denotes the activation function of the ith hidden layer; $\boldsymbol{h}^{(L)}$ denotes the output of the encoder.

The red arrows in Fig. 1 stand for the process of decoding, which is composed of several hidden layers and an output layer. It has the opposite purpose with the encoder, which inputs the deep compressed features to reconstruct the original data to the greatest extent. Correspondingly, the forward calculation in the decoder can be expressed as

$$\begin{aligned}\mathbf{h}^{(i)} &= \sigma^{(i)}\left(W^{(i)}\mathbf{h}^{(i-1)}+\mathbf{b}^{(i)}\right), i=L+1,L+2,\ldots,o-1\\ \tilde{\mathbf{x}} &= \sigma^{(o)}\left(W^{(o)}\mathbf{h}^{(o-1)}+\mathbf{b}^{(o)}\right)\end{aligned} \tag{2}$$

where $\tilde{\boldsymbol{x}}\epsilon\boldsymbol{R}^{d_x}$ is the output of the DAE, which has the same size as the input variables; the superscript o denotes the output layer; $\sigma^{(o)}(\cdot)$ denotes the activation function in the output layer. Especially, if DAE satisfies $d_{(i)} = d_{(o-i)}, i = 1, 2, \ldots, L$, then the number of neurons in the encoder and decoder is the same but in reverse order.

The parameters to optimize in DAE can be described as $\{W^{(i)}, \boldsymbol{b}^{(i)}\}_{i=1,2,\ldots,L,L+1,\ldots o}$, which are the weight matrices and bias vectors in each layer. The objective of DAE is to minimize the error between the input and its reconstruction. Usually, the mean squared error is applied to learn the network parameters in DAE

$$J\left(W^{(1)},\mathbf{b}^{(1)},\ldots,W^{(o)},\mathbf{b}^{(o)}\right)=\sum_{n=1}^{N}\left\|\tilde{\mathbf{x}}_n-\mathbf{x}_n\right\|_2^2/N \tag{3}$$

where $||\cdot||_2^2$ denotes the Euclidean norm; N represents the number of samples inputted into the DAE at one time. In batch training, N is the batch size. After calculating the error using (3), the backpropagation (BP) algorithm is employed to propagate the error to each layer of the DAE. Then, the parameters of the DAE will be updated iteratively through optimization algorithms, such as stochastic gradient descent [24], Adam [25], RMSprop [26], etc. After training, the DAE will precisely reconstruct the original data with a minor error.

B. DAE for Imputation Based on Supervised Learning

DAEs can be used for missing value imputations. The imputation process can be implemented differently, depending on performed strategies. The strategy of most published works for imputations [27]–[29] is based on supervised learning, in which only complete data are used to train the imputation model. Missing values in incomplete data are imputed with their reconstruction during the testing phase. In the training phase of supervised DAE, missing values are generated from the complete data by random dropping. Then, generated incomplete data are fed into DAE, and their complete original data are used to calibrate the reconstruction. The process of supervised learning-based DAE for imputation is shown in Fig. 2.

In Fig. 2, $\boldsymbol{x}$ represents the generated incomplete sample; symbols "?" indicate locations of missing values; and the marker vector $\boldsymbol{m}$ indicates whether the values in the incomplete sample are missed. It can be defined as

$$m_j=\begin{cases}1, & \text{if } x_j \text{ is missed}\\ 0, & \text{otherwise.}\end{cases} \tag{4}$$

The function init$(\cdot)$ can be zero, Gaussian noise, the mean of variables, etc., which generates initial values to replace non-computable missing values in the input. After that, forward propagations are performed to obtain reconstructed values. Through equations, it can be expressed by

$$\begin{aligned}\bar{\mathbf{x}} &= (1-\mathbf{m})\odot\mathbf{x}+\mathbf{m}\odot\text{init}(\mathbf{x})\\ \tilde{\mathbf{x}} &= \text{DAE}(\bar{\mathbf{x}})\end{aligned} \tag{5}$$

where $\widehat{\boldsymbol{x}}$ is an initialized sample as a complete sample feeding to DAE; the notation $\odot$ denotes elementwise multiplication; DAE$(\cdot)$ refers to the forward propagation of DAE; and $\widetilde{\boldsymbol{x}}$ is the reconstruction. The error between the original value $\boldsymbol{x}_{\text{real}}$ and its reconstruction $\widetilde{\boldsymbol{x}}$ is used to learn the parameters. The loss function of supervised DAEs for imputation can be represented as

$$J_{\text{sup}}\left(W^{(1)},\mathbf{b}^{(1)},\ldots,W^{(o)},\mathbf{b}^{(o)}\right)=\sum_{n=1}^{N}\left\|\tilde{\mathbf{x}}_n-\mathbf{x}_{\text{real},n}\right\|_2^2/N. \tag{6}$$

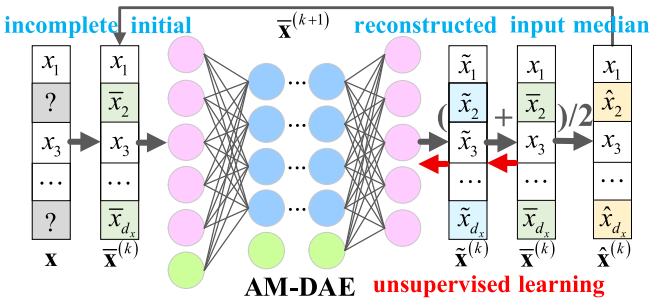


Fig. 3. AM-DAE for recursive imputation based on unsupervised learning.

After sufficiently training the DAE, the missing values in the incomplete data set can be predicted by forward propagation. Hence, the imputed data can be represented by

$$\mathbf{x}_{\text{impu}} = \hat{\mathbf{x}} = (1 - \mathbf{m}) \odot \mathbf{x} + \mathbf{m} \odot \tilde{\mathbf{x}} \tag{7}$$

where $\widehat{\boldsymbol{x}}$ denotes the complete sample. The first term of the right-hand side of (7) satisfies $(1 - \boldsymbol{m}) \odot \boldsymbol{x} = (1 - \boldsymbol{m}) \odot \widehat{\boldsymbol{x}} = (1 - \boldsymbol{m}) \odot \widehat{\boldsymbol{x}}$.

III. Adaptive-Learned Median-Filled Deep Autoencoder

Although supervised DAE is a straightforward and efficient imputation approach, it is limited in dealing with incomplete data that occupy the majority of the entire data set. The limited sample size and information will cause the model to overfit and learn a biased data distribution. Also, if there is a significant distinction between the complete and incomplete data distributions, the imputation accuracy of the supervised DAE will be hard to ensure. In addition, the single prediction used in supervised DAE is sensitive to initial values.

In this section, an unsupervised learning-based AM-DAE for imputations is developed. It avoids the above shortcomings, also improving the imputation performance. A dynamic extension of the AM-DAE is also introduced by considering the time-series characteristics of the process data.

A. Adaptive-Learned Median-Filled Deep Autoencoder

AM-DAE is a deep network designed to improve the accuracy of unsupervised imputation. The iterative median imputation keeps its previous imputed values transmissible. The adaptive learning strategy determines the learning priority of the model. Also, stacked dynamic samples allow the model to impute frequently and learn efficiently.

In AM-DAE, all the nonmissing values from complete and incomplete samples will be used to learn the model parameters, ensuring that the data information can be fully utilized. The imputations of missing values and the training of model parameters are performed concurrently. After each forward propagation, the missing values are recursively replaced by the median of input and its reconstruction. Once the model has finished training, the missing values are simultaneously accomplished in their imputations.

Fig. 3 depicts the primary procedure of AM-DAE for imputation. We can see that the input as a label is applied to reduce reconstruction errors and learn appropriate parameters. The median is used as imputed values to continuously update the initial values in the incomplete input. In fact, parameter and initial value and updates are mutually reinforcing. Changes in initial value will produce model deviations, while parameter updating promotes reconstruction deviations. Finally, they will converge to reach an equilibrium. Through equations, the imputation procedure shown in Fig. 3 can be defined by

$$\begin{aligned}
\bar{\mathbf{x}}^{(1)} &= (1 - \mathbf{m}) \odot \mathbf{x} + \mathbf{m} \odot \text{init}(\mathbf{x}) \\
\tilde{\mathbf{x}}^{(k)} &= \text{AM} - \text{DAE}\left(\bar{\mathbf{x}}^{(k)}\right), k = 1, 2, \ldots \\
\mathbf{x}_{\text{median}}^{(k)} &= \left(\bar{\mathbf{x}}^{(k)} + \tilde{\mathbf{x}}^{(k)}\right)/2, k = 1, 2, \ldots \\
\hat{\mathbf{x}}^{(k)} &= (1 - \mathbf{m}) \odot \bar{\mathbf{x}}^{(k)} + \mathbf{m} \odot \mathbf{x}_{\text{median}}^{(k)}, k = 1, 2, \ldots \\
\bar{\mathbf{x}}^{(k+1)} &= \hat{\mathbf{x}}^{(k)}, k = 1, 2, \ldots
\end{aligned} \tag{8}$$

where $\text{AM} - \text{DAE}(\cdot)$ represents the forward propagation of AM-DAE; $\boldsymbol{x}_{\text{median}}^{(k)}$ is the median of the input and its reconstruction at the kth iteration; $\widehat{\boldsymbol{x}}^{(k)}$ is the imputed sample that the missing values are replaced by $\boldsymbol{x}_{\text{median}}^{(k)}$; and $\widehat{\boldsymbol{x}}^{(k+1)}$ is the initialized input in the $(k+1)$th iteration, which equals $\widehat{\boldsymbol{x}}^{(k)}$.

The reason to use the median instead of the reconstruction to update the initialized input is that the median can memorize the previous imputations. This strategy can alleviate the sharp change in the imputed value of adjacent iterations and make the imputation operation heritable. This idea is also adopted in several optimization problems. For example, momentum is a popular optimization algorithm for parameter learning in deep networks. It updates the current gradient by referring to the previous gradient, yielding lower oscillations and smoother updates in parameters. Another example is the step of individual selection in swarm intelligence algorithms, in which half of the best individuals are selected for the next generation. This strategy preserves outstanding individuals and accelerates the searching for global optimization. In this article, the median can be treated as 50% weighting of the previous and current reconstructed values. It speeds up the training process and improves imputation accuracy, verified in simulations.

In addition, AM-DAE employs an adaptive learning strategy considering that unsupervised imputations can be analogous to multitask learning, that is, reconstruction of nonmissing values and imputation of missing values. Undoubtedly, it is not easy to impute missing values correctly until the model has learned the data distribution well. There should be a priority for what AM-DAE should focus on during the different training phases. In the early training phase, we hope that AM-DAE can pay more attention to the reconstruction learning for nonmissing values and weaken imputation learning. As the iteration proceeds, the attention devoted to minimizing the reconstruction and imputation errors will gradually equalize and eventually reach parity. Hence, an adaptive loss function in AM-DAE is designed to realize the above function, thereby providing a more reasonable update of network parameters. The adaptive loss function can be defined by

$$\begin{aligned}
J_{adp} &= \sum_{n=1}^{N} \left\| \left(\alpha^{(k)}(1 - \mathbf{m}_n) + \beta^{(k)}\mathbf{m}_n\right) \odot \left(\tilde{\mathbf{x}}_n^{(k)} - \bar{\mathbf{x}}_n^{(k)}\right) \right\|_2^2 / N \\
\alpha^{(k)} &= 2(1 - 0.5k/\text{epoch}) \\
\beta^{(k)} &= k/\text{epoch}
\end{aligned} \tag{9}$$

where $\boldsymbol{m}_n$ indicates the location of missing values in the nth sample; $\alpha^{(k)}$ and $\beta^{(k)}$ are the adaptive coefficients of the reconstruction error and the imputation error, respectively, which are related to the number of iterations k; and *epoch* denotes the maximum number of iterations. With the increase of k, $\alpha^{(k)}$ will change from 2 to 1, and $\beta^{(k)}$ will vary from 0 to 1.

The update of network parameters is performed simultaneously with the imputation of missing values. The training process of AM-DAE can be described as follows.

1) When $k = 1$, the missing values in the input sample are replaced by initial values generated from init$(\cdot)$; otherwise, replaced by the imputed values in the previous iteration.
2) By forward propagation, the reconstruction can be obtained. Then, the median of the input and its reconstruction is calculated by (8).
3) The error between the input and its reconstruction is obtained by (9). The network parameters are updated using a selected optimization algorithm.
4) Missing values in incomplete samples are imputed by the median, also utilized as the initial value for the input in the next iteration. If the data set has not been entirely looped, go to the following sample; otherwise, move to the next iteration and go to step 1).

Correspondingly, the pseudocode of AM-DAE is indicated in Algorithm 1. The complete data set will be acquired as the AM-DAE finishes its training. The final imputed values can be represented by

$$\mathbf{x}_{\text{impu}} = \hat{\mathbf{x}}^{(\text{epoch})}. \tag{10}$$

Although Algorithm 1 provides an imputation procedure for AM-DAE, it suffers from the problem that the number of imputations is much less than that of parameter learning. Every sample is updated only once for each iteration of the training set. However, network parameters can be learned T/N times, where T and N represent the sample size of the training set and batch size, respectively. The unbalanced number of updates results in overlearning for nonmissing values and under learning for missing values. Hence, an expanded imputation algorithm of AM-DAE is proposed to deal with time-series missing values in the following section, which can alleviate the unbalanced update frequency in Algorithm 1.

Algorithm 1: AM-DAE

Input: Missing data set X and its missing position matrix M, maximum number of iterations *epoch*, batch size N

Output: Imputed data set $\hat{x}^{(epoch)}$

$\bar{X}^{(1)} = (1 - M) \odot X + M \odot$ init (X);
for $k \leftarrow 1$ **to** *epoch* **do**
  $\alpha^{(k)} = 2(1 - 0.5k/epoch)$;
  $\beta^{(k)} = k/epoch$;
  Randomly shuffle data set $(\bar{X}^{(k)}, M)$;
  $p = 1$;
  while $(\bar{X}^{(k)}, M)$ *has not been completely looped* **do**
    Draw N samples from $(\bar{X}^{(k)}, M)$, denoted as $\{(\bar{X}_n^{(k)}, \mathbf{m}_n)\}_{n=p}^{p+N-1}$;
    $loss = 0$;
    for $n \leftarrow p$ ***to*** $p + N - l$ **do**
      $\tilde{\mathbf{x}}_n^{(k)} = \text{AM} - \text{DAE}(\bar{\mathbf{x}}_n^{(k)})$;
      $J_{adp,n} = \|(\alpha^{(k)}(1 - \mathbf{m}_n) + \beta^{(k)}\mathbf{m}_n) \odot (\tilde{\mathbf{x}}_n^{(k)} - \bar{\mathbf{x}}_n^{(k)})\|_2^2$;
      $loss = loss + J_{adp,n}$;
      $\mathbf{x}_{median,n}^{(k)} = (\bar{\mathbf{x}}_n^{(k)} + \tilde{\mathbf{x}}_n^{(k)})/2$;
      $\hat{\mathbf{x}}_n^{(k)} = (1 - \mathbf{m}_n) \odot \bar{\mathbf{x}}_n^{(k)} + \mathbf{m}_n \odot \mathbf{x}_{median,n}^{(k)}$;
      $\bar{\mathbf{x}}_n^{(k+1)} = \hat{\mathbf{x}}_n^{(k)}$;
    end
    $loss = loss/N$;
    Back propagate loss;
    $p = p + N$;
  end
end

B. Imputations of Time-Series Missing Values Using AM-DAE

Consider stacking Δt samples of consecutive sampling intervals as a dynamic sample, denoted as

$$\mathbf{x}_{\Delta t}(t) = \left[\mathbf{x}^{\mathrm{T}}(t + 1 - \Delta t) \quad \mathbf{x}^{\mathrm{T}}(t + 2 - \Delta t) \quad \cdots \quad \mathbf{x}^{\mathrm{T}}(t)\right]^{\mathrm{T}} \tag{11}$$

where $\boldsymbol{x}(t) \epsilon \boldsymbol{R}^{d_v}$ represents the tth collected sample; d_v denotes the number of process variables; and the superscript T represents the transpose operation. Then, a dynamic sample set with batch size N can be denoted as

$$X_N = \{\mathbf{x}_{\Delta t}(t_1), \mathbf{x}_{\Delta t}(t_2), \ldots, \mathbf{x}_{\Delta t}(t_N)\}. \tag{12}$$

One should note that in random batch training, t_n and t_{n+1} can be two nonadjacent sampling moments. Assume that the total number of sampling is T. Then, $T + 1 - \Delta t$ dynamic samples can be generated via (11).

Specifically, the pseudocode of AM-DAE for imputing time-series missing data is illustrated in Algorithm 2, in which the subscript N represents a batch of dynamic samples; and $\hat{X}_{N,n}^{(k)}$ stands for the nth reconstructed dynamic sample in the batch at the kth iteration.

Fig. 4 presents an example to visualize the training process of Algorithm 2. It can be seen that there are two training batches in Fig. 4. The batch size N of dynamic sample is equal to two, and each dynamic sample is composed of two stacked samples ($\Delta t = 2$). The small squares in each colored box constitute a dynamic sample, which contains $d_x = \Delta t \cdot d_v$ values. The update step of stacked dynamic samples can be described as follows.

1) The missing values in the original data are replaced with the initial values generated from the function init$(\cdot)$.
2) For each batch, N dynamic samples are chosen randomly from the dynamic sample set. They are converted into vectors to be able to be fed into AM-DAE.
3) Through forward propagation, the corresponding reconstruction $\tilde{X}_N$, median $X_{N,\text{median}}$, and imputation $\hat{X}_N$ can be obtained from the input $\hat{X}_N$ using (8). Then, each

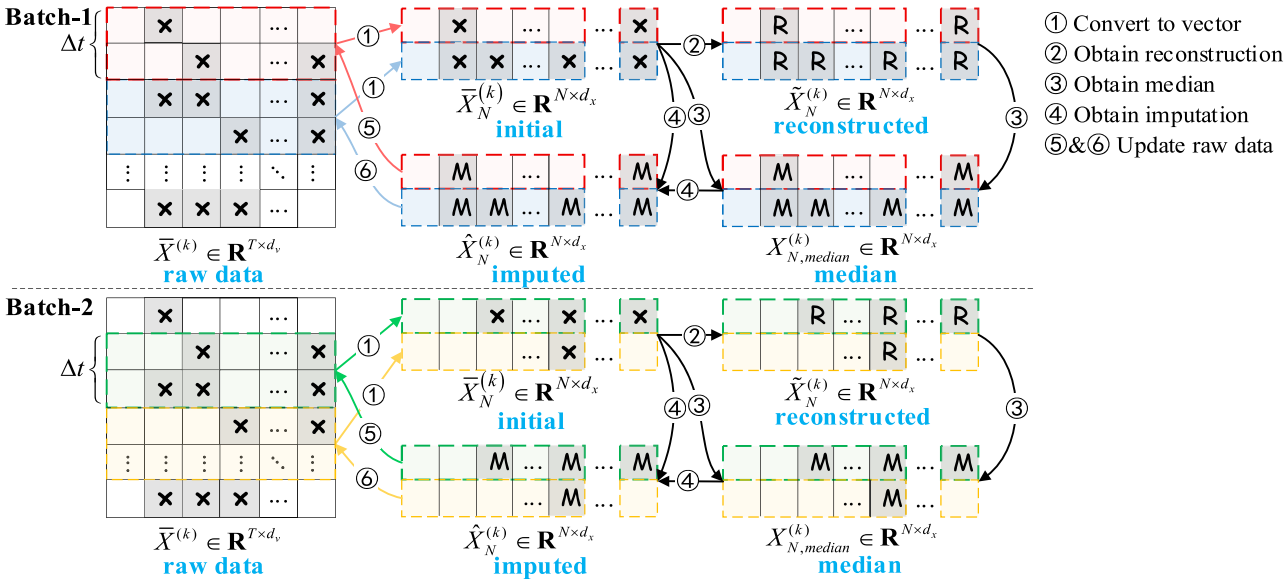


Fig. 4. Example of the imputation in two batches of dynamic samples using Algorithm 2.

Algorithm 2: AM-DAE for Time-Series Missing Data Imputation

Input: Missing data set X and its missing position matrix M, maximum number of iterations *epoch*, batch size N, number of stacked samples Δt, sample size for training T

Output: Imputed data set $\hat{X}^{(epoch)}$

$\bar{X}^{(1)} = (1 - M) \odot X + M \odot \text{init}(X)$
for $k \leftarrow 1$ **to** *epoch* **do**
  $\alpha^{(k)} = 2(1 - 0.5k/epoch)$;
  $\beta^{(k)} = k/epoch$;
  Get a randomly shuffled integer order $S = \{s_j\}_{j=1}^{T-\Delta t+1}$ ranging from Δt to T;
  $p = 1$;
  while *S has not been completely looped* **do**
    Draw N integers from S, denoted as $\{t_n\}_{n=1}^{N} = \{s_j\}_{j=p}^{p+N-1}$;
    Get stacked dynamic samples from $\bar{X}^{(k)}$ based on the integers, denoted as $\bar{X}_N^{(k)} = \{\bar{\mathbf{x}}_{\Delta t}^{(k)}(t_1), \bar{\mathbf{x}}_{\Delta t}^{(k)}(t_2), \ldots, \bar{\mathbf{x}}_{\Delta t}^{(k)}(t_N)\}$;
    Calculate reconstruction, median, and imputation matrix $(\tilde{X}_N^{(k)}, X_{N,median}^{(k)}, \hat{X}_N^{(k)})$;
    Calculate loss and perform backpropagation;
    for $n \leftarrow 1$ **to** N **do**
      $[\bar{\mathbf{x}}^{(k+1)}\mathrm{T}(t_n + 1 - \Delta t) \cdots \bar{\mathbf{x}}^{(k+1)}\mathrm{T}(t_n)] = \hat{X}_{N,n}^{(k)}$
    end
    $p = p + N$;
  end
end

imputed values will replace the corresponding missing values in the original data as their initial values in the subsequential iteration.

4) Based on (9), the loss is calculated. BP is performed to update parameters of AM-DAE.
5) By looping 2)–4), the entire training set of data can be looped once.
6) By looping 5) epoch times, AM-DAE is well trained. The final imputed data set is obtained via (10).

By stacking data as dynamic samples, the number of imputations of an original sample in a loop of the training set can be increased from one to a maximum of Δt times. This dramatically improves the imputation frequency of the original data. In this way, frequent imputation of missing values will cause fluctuations in reconstruction errors, forcing the network parameters to be frequently updated, thus improving learning efficiency.

IV. Case Studies

In this section, the imputation performance of AM-DAE is verified through the TE process and an actual chemical process. The testing results of these two cases illustrate the effectiveness of AM-DAE. Before that, the generation of industrial-type missing values and criteria for evaluating the imputation performance are introduced. The code of this simulation can be found in https://github.com/fuzimaoxinan/Pytorch-Deep-Neural-Networks.

A. Industrial-Type Missing Values and Performance Indicators of Imputations

In most studies, missing values are introduced randomly. However, it is inconsistent with the missing forms in the real industrial world. In actual industrial processes, the appearance of missing values is not random but regular. They usually occur in fixed intervals and continuous periods.

The missing patterns of these two industrial-type values are shown in Fig. 5, in which the vertical direction of the matrix represents the time scale; the horizontal direction represents

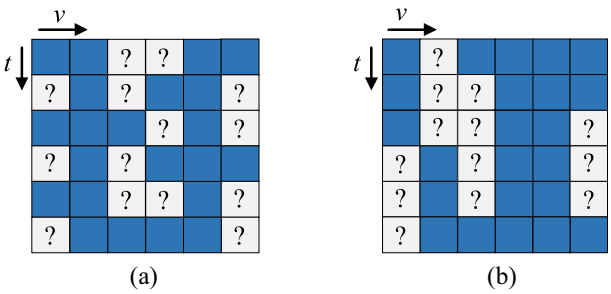


Fig. 5. Two common patterns of industrial-type missing values. (a) Fixed intervals. (b) Continuous periods.

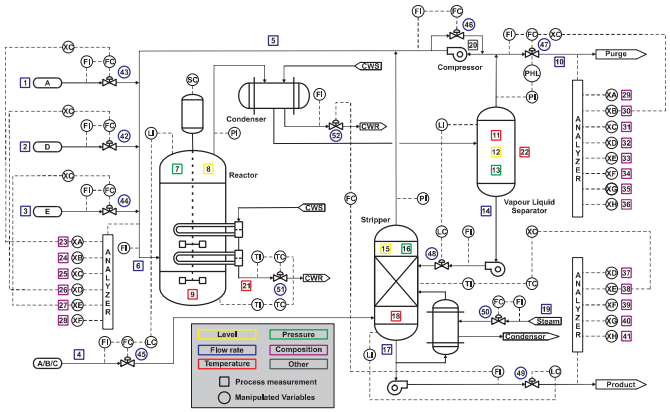


Fig. 6. Benchmarked TE process [31].

different variables; and symbols "?" indicate locations of missing values. In general, missings in fixed intervals are caused by multiple sampling rates. There are three different sampling rates illustrated in Fig. 5(a). Assume the sampling interval for the second and fifth variables is τ. Then, the first and fourth variables collect a value at every 2τ. Thereby, the sampling interval for the third and sixth variables is 3τ. Regarding the second form of missing depicted in Fig. 5(b), it is usually caused by packet loss. Due to sensor failure, transmission stoppage, reception abnormalities, and other reasons, data are unavailable for continuous periods.

In order to evaluate the imputation performance, the root mean square error (RMSE) and mean absolute percentage error (MAPE) are defined, satisfying

$$\text{RMSE} = \sqrt{\frac{1}{N_M}\sum_{t,v} M_{t,v}\left(\hat{X}_{t,v} - X_{real,t,v}\right)^2}$$
$$\text{MAPE} = \frac{1}{N_M}\sum_{t,v} M_{t,v}\left|\frac{\hat{X}_{t,v} - X_{real,t,v}}{X_{real,t,v}}\right| \times 100\% \quad (13)$$

where N_M is the total number of missing values in the data set; $M_{t,v}$ indicates whether the vth variable of the tth sample is missing, which takes one if missing; and $\hat{X}$ and X_{real} are the imputed data set and the complete data set, respectively. Note that when $X_{\text{real},t,v}$ is zero or close to zero, the MAPE index becomes meaningless. Hence, to avoid this problem, the data will be normalized to a range of 0.1–0.9 before being used for training.

B. TE Process

TE process [30] is a widespread benchmark in process control and fault diagnosis, consisting of a reactor, a condenser, a compressor, a separator, and a stripper. Its flow chart can be described by Fig. 6. TE process measures 52 variables every 3 min under 22 different conditions, including a normal and 19 faulty status. The data sets used in this simulation are offered by this article [31], available in https://github.com/camaramm/tennessee-eastman-profBraatz. For consistency with most studies, only 22 continuous measures and 11 manipulated variables were selected for modeling. Also, the data sets with the categories of "Fault 03," "Fault 09," and "Fault 15" were not considered in this study [12], [32]. In this way, $500+480\times 18 = 9140$ original samples were obtained. By setting the length of stacking $\Delta t = 16$, 8855 stacked dynamic samples were generated.

TABLE I
GENERATING INCOMPLETE DATA SETS FROM TE WITH DIFFERENT MISSING RATES

Incomplete data set	1/2 Sampling Rate	1/3 Sampling Rate	Continuous missing
Data (10%)	3Vs (50%)	2Vs (67%)	1V (46%)
Data (20%)	5Vs (50%)	4Vs (67%)	3Vs (47%)
Data (30%)	7Vs (50%)	6Vs (67%)	4Vs (60%)
Data (40%)	8Vs (50%)	9Vs (67%)	6Vs (53%)
Data (50%)	10Vs (50%)	11Vs (67%)	8Vs (52%)
Data (60%)	10Vs (50%)	11Vs (67%)	12Vs (62%)
Data (70%)	9Vs (50%)	10Vs (67%)	14Vs (85%)

Based on the missing patterns of industrial-type data, seven incomplete data sets with different missing rates were generated, which are listed in Table I. One can see that three types of missing patterns were adopted, where "1/2 or 1/3 sampling rate" refers to sampling data at twice or three times basic intervals τ; "Continuous missing" means that data are missing continuously over a period of time. Furthermore, the missing variables were only generated in specific variables, randomly selected from 33 variables. Table I displays the number of selected variables and their missing rate. For example, "Data (10%)" represents a total of 10% values in the original data set are missing. "3Vs (50%)" means that three variables are missing at a "1/2 sampling rate," and incomplete values account for 50% of the total data values. For the first row in Table I, it satisfies $33 \times 10\% = 3 \times 50\% + 2 \times 67\% + 1 \times 46\%$.

The seven generated incomplete sets were employed to train imputation methods, and the original complete set was utilized to test the imputation performance of methods. As listed in Table II, five different unsupervised approaches were built, in which "DAE" denotes the basic DAE without median imputation and adaptive loss; "Epoch" represents the number of iterative training; "Model structure" denotes the number of neurons selected in each layer. For example, the structure of DAE shows that the number of neurons in its input and output layer is $16 \times 33 = 528$. It has three hidden layers, all having 264 neurons. In particular, the two structures listed in GAIN

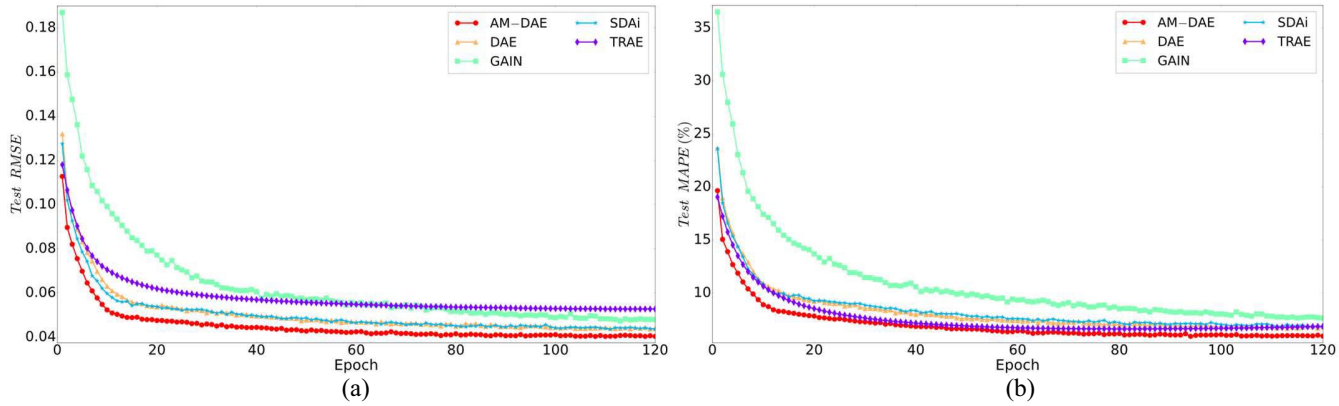


Fig. 7. Comparison of RMSE and MAPE of five methods on "Data (20%)" of the TE process. (a) RMSE. (b) MAPE(%).

TABLE II
STRUCTURE AND HYPERPARAMETERS SETTING OF IMPUTATION MODELS IN TE PROCESS

Networks	Model structure	Other hyperparameters
TRAE [21]	(528, 264, 528)	{Func =LeakyReLU, Epoch = 120, Dropout = 0.1, Batch size = 32, Learning rate = 1e-4}
DAE	(528, 264, 264, 264, 528)	
SDAi [33]	(528, 264, 264, 264, 528)	
AM-DAE	(528, 264, 264, 264, 528)	
GAIN [17]	(528, 264, 264, 264, 528)	
	(528, 264, 264, 264, 528)	

represent the number of neurons in the generator and discriminator, respectively. In Table II, "Func" denotes the activation function used in five models, satisfying

$$\text{LeakyRELU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0.01x, & \text{otherwise.} \end{cases} \tag{14}$$

In addition, "Dropout" is a technique to prevent models from overfitting. It randomly drops activations $\boldsymbol{h}$ with a probability of p_{dropout}, which satisfies

$$\text{dropout}(h_j) = \begin{cases} 0, & \text{if } \eta_j < p_{\text{dropout}} \\ \frac{h_j}{1-p_{\text{dropout}}}, & \text{otherwise} \end{cases} \tag{15}$$

where random number η_j obeys a (0, 1) uniform distribution.

The imputation results for the five models are shown in Table III, in which the mean and variance of RMSE and MAPE for ten replicate experiments are displayed. It can be seen that the imputation performance of AM-DAE is the best among the five methods for different missing rates. To demonstrate the advantages of AM-DAE in terms of imputation efficiency and accuracy, Fig. 7 shows the RMSE and MAPE test results for the five models on "Data (20%)." Each point in the figure represents the test results after each loop of training the data set. One can see that AM-DAE has faster learning efficiency and better imputation performance. In detail, the final imputation curve of AM-DAE for the 12 missing variables of "Data (20%)" is presented in Fig. 8, in which the Variable 2, 3, 4, 10, and 25 are missed causing by "1/2 sampling rate"; Variable 1, 9, 11, and 19 are generated at "1/3 sampling rate"; and Variable 16, 18, and 26 are continuously missing values from a certain point. In the subview of each variable, the original/real values indicated by the red line are sorted from small to large. Their corresponding imputed values are represented by blue scatter points. It can be seen that in most cases, the trend of the imputed values remains consistent with the real values. The results in Variables 2 and 9 are relatively poor, in which most of the imputed values are concentrated near the mean. It implies that those variables provide insufficient real values for learning. In this case, even replacing the missing values with the initial mean value yields a small loss. The model falls into a local minimum.

C. Hydrocracking Process

Hydrocracking is an essential ring of the oil refining process. It filters sulfur and nitrogen impurities in crude oil and cracks them into light-molecular product oils. In Fig. 9, the flowchart of an actual petrochemical hydrocracking process in China is shown. It mainly consists of two reactors, four separators, and a stripper.

In this study, 14 855 samples were collected for missing value imputations. Through careful selection, 61 process variables were used for training and testing. Similarly, five industrial-type incomplete data sets with different missing rates were generated based on the missing patterns of "1/2 sampling rate," "1/3 sampling rate," and "continuous missing." In Table IV, the network structure and hyperparameters of five imputation models were designed, the meaning of which can refer to the description of Table II. Furthermore, the number of stacked samples was set to 16, thus, generating 13 067 available dynamic samples.

As a result of the test, the mean and variance of RMSE and MAPE for the five replicate experiments are presented in Table V. The values in parentheses represent the variance. From the results, we can infer that AM-DAE is superior in imputation performance in most cases. Compared with the other four unsupervised methods, AM-DAE can achieve lower RMSE and MAPE in most incomplete data sets. In addition, it is noted that the RMSE/MAPE of these five methods on the incomplete data set with a missing rate of 50% is lower than the case of a missing rate of 40%. This is because the variables used to generate the missing data set were randomly

TABLE III
MEAN AND VARIANCE (IN PARENTHESES) OF RMSE AND MAPE (%) FOR TEN REPLICATE EXPERIMENTS OF THE TE PROCESS

Incomplete data set	RMSE					MAPE (%)				
	TRAE	DAE	SDAi	AM-DAE	GAIN	TRAE	DAE	SDAi	AM-DAE	GAIN
Data (10%)	0.0480 (3e-5)	0.0297 (4e-8)	0.0306 (8e-7)	**0.0270** (9e-8)	0.0352 (5e-7)	6.06 (2e-2)	5.49 (1e-3)	5.64 (6e-3)	**4.89** (4e-4)	6.73 (1e-2)
Data (20%)	0.0528 (4e-5)	0.0438 (9e-8)	0.0436 (2e-7)	**0.0404** (5e-8)	0.0474 (2e-8)	6.83 (7e-2)	6.71 (1e-2)	6.86 (5e-3)	**5.97** (1e-4)	7.69 (3e-4)
Data (30%)	0.0607 (9e-6)	0.0533 (1e-8)	0.0532 (3e-7)	**0.0508** (1e-8)	0.0624 (4e-8)	8.33 (3e-2)	8.72 (7e-2)	8.67 (5e-2)	**7.99** (2e-4)	10.32 (5e-2)
Data (40%)	0.0646 (1e-6)	0.0570 (6e-7)	0.0582 (3e-7)	**0.0534** (8e-8)	0.0658 (6e-7)	9.57 (2e-2)	8.71 (1e-2)	8.63 (7e-3)	**7.74** (1e-3)	10.51 (6e-3)
Data (50%)	0.0542 (7e-5)	0.0533 (2e-8)	0.0542 (5e-7)	**0.0507** (7e-8)	0.0613 (3e-8)	7.81 (1e-2)	7.65 (3e-2)	7.68 (1e-3)	**7.06** (2e-4)	9.24 (3e-2)
Data (60%)	0.0735 (2e-5)	0.0623 (3e-8)	0.0659 (9e-7)	**0.0612** (7e-8)	0.0692 (7e-8)	11.81 (8e-2)	9.95 (2e-2)	10.15 (5e-3)	**9.59** (8e-4)	11.69 (2e-2)
Data (70%)	0.0910 (6e-5)	0.0766 (6e-8)	0.0803 (8e-7)	**0.0742** (2e-8)	0.0808 (1e-7)	16.17 (4e-2)	12.60 (1e-2)	12.95 (2e-2)	**12.15** (4e-3)	14.13 (9e-3)

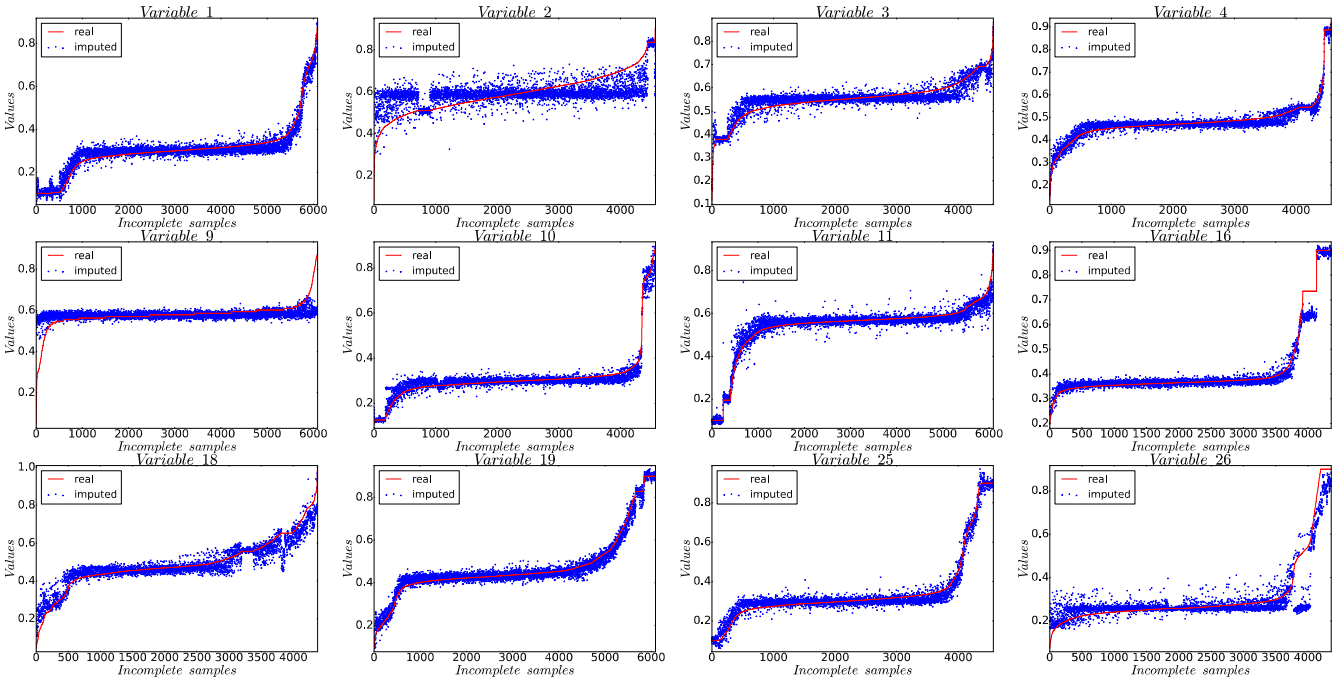


Fig. 8. Imputation curve of AM-DAE on "Data (20%)" of the TE process.

TABLE IV
NETWORK STRUCTURE AND HYPERPARAMETERS SETTING OF IMPUTATION MODELS IN HYDROCRACKING PROCESS

Networks	Model structure	Other hyperparameters
TRAE [21]	(976, 488, 976)	{Func =LeakyReLU, Epoch = 120, Dropout = 0.1, Batch size = 16, Learning rate = 1e-4}
DAE	(976, 488, 488, 488, 976)	
SDAi [33]	(976, 488, 488, 488, 976)	
AM-DAE	(976, 488, 488, 488, 976)	
GAIN [17]	(976, 488, 488, 488, 976)	
	(976, 488, 488, 488, 976)	

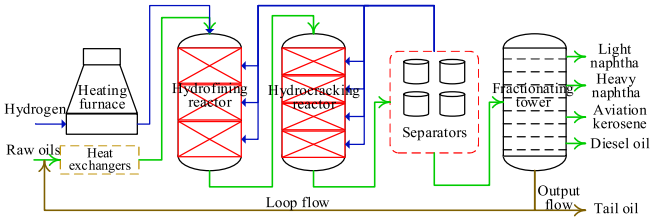


Fig. 9. Actual hydrocracking process [34].

selected, with some of them being easier to impute. Therefore, this phenomenon is justified in this study.

In Fig. 10, the imputed values of each missing variable in the "Data (10%)" data set of hydrocracking are provided by AM-DAE. One can see that the values imputed by AM-DAE are almost close to the real value line, and only a tiny percentage of the imputed values deviate. These inaccurate imputation values have a notable abrupt change in the graph, which may be caused by outliers.

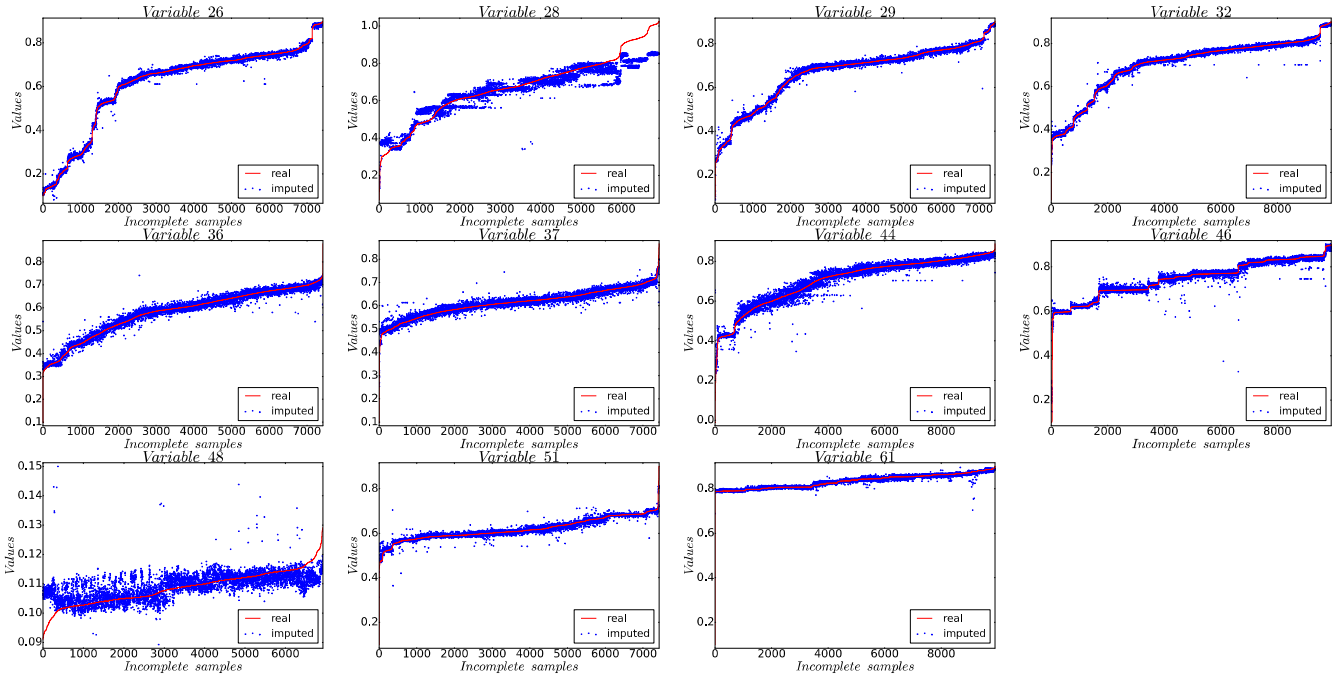


Fig. 10. Imputation curve of AM-DAE on "Data (10%)" of the hydrocracking process.

TABLE V
MEAN AND VARIANCE (IN PARENTHESES) OF RMSE AND MAPE (%) FOR FIVE REPEATED EXPERIMENTS ON THE HYDROCRACKING IMPUTATION

Incomplete data set	RMSE					MAPE (%)				
	TRAE	DAE	SDAi	AM-DAE	GAIN	TRAE	DAE	SDAi	AM-DAE	GAIN
Data (10%)	0.0233 (5e-5)	0.0231 (3e-7)	0.0237 (2e-7)	**0.0212** (9e-8)	0.0325 (6e-8)	2.75 (1e-2)	2.29 (3e-3)	2.42 (9e-3)	**1.94** (4e-4)	3.86 (5e-2)
Data (20%)	0.0382 (8e-5)	0.0324 (4e-8)	0.0329 (1e-7)	**0.0313** (7e-8)	0.0454 (1e-7)	4.01 (8e-2)	3.73 (6e-2)	3.71 (1e-3)	**3.41** (5e-4)	5.59 (4e-3)
Data (30%)	0.0536 (3e-6)	0.0477 (6e-7)	**0.0460** (6e-7)	0.0462 (8e-8)	0.0571 (4e-7)	5.07 (4e-2)	4.48 (1e-2)	4.14 (8e-2)	**4.07** (6e-4)	6.47 (1e-3)
Data (40%)	0.0614 (1e-5)	0.0566 (7e-8)	0.0548 (2e-7)	**0.0545** (3e-8)	0.0639 (2e-7)	10.28 (2e-2)	8.68 (6e-2)	7.89 (7e-3)	**7.80** (4e-3)	9.93 (9e-3)
Data (50%)	0.0598 (7e-5)	0.0477 (1e-8)	0.0448 (8e-7)	**0.0445** (2e-8)	0.0577 (1e-8)	6.75 (2e-2)	4.64 (8e-2)	4.13 (9e-3)	4.12 (3e-4)	5.87 (5e-2)

On the whole, the superior imputation capability of AM-DAE is verified by several incomplete data sets of TE and hydrocracking processes. It is more efficient and accurate than the other four unsupervised algorithms.

V. CONCLUDING REMARKS

In this article, a novel AM-DAE is proposed for time-series missing data imputation. In AM-DAE, the network training and sample imputation are performed simultaneously. Based on unsupervised learning, missing values are recursively imputed by the median of the input and its reconstructed values after each forward propagation. An adaptive loss function is also used to update the model parameters, allowing AM-DAE to focus more on reconstruction learning in the early training phase and reaching a tradeoff between reconstruction and imputation learning at the end of the training. Moreover, AM-DAE is extended to deal with time-series missing values. The imputation frequency and learning efficiency are significantly increased. In the experimental part, industrial-type missing patterns, including missing in fixed intervals and continuous periods, were applied to generate incomplete data sets based on different missing rates. Two examples of chemical processes, namely, TE and hydrocracking processes, verified the imputation performance of AM-DAE. It was demonstrated that AM-DAE outperforms other advanced unsupervised imputation methods, presenting lower RMSE, less MAPE, and fast learning efficiency. In most cases, the imputed values provided by AM-DAE can keep up with the trend of the real value curve.

REFERENCES

[1] Q. Ma *et al.*, "End-to-end incomplete time-series modeling from linear memory of latent variables," *IEEE Trans. Cybern.*, vol. 50, no. 12, pp. 4908–4920, Dec. 2020.

[2] H. Chen and B. Jiang, "A review of fault detection and diagnosis for the traction system in high-speed trains," *IEEE Trans. Intell. Transp.*, vol. 21, no. 2, pp. 450–465, Feb. 2020.

[3] K. Liu, Y. Li, J. Yang, Y. Liu, and Y. Yao, "Generative principal component thermography for enhanced defect detection and analysis," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 10, pp. 8261–8269, Oct. 2020.

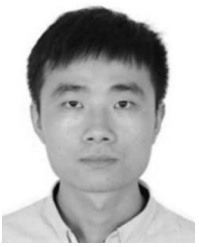
[4] J. Chen, Z. Liu, H. Wang, A. Núñez, and Z. Han, "Automatic defect detection of fasteners on the catenary support device using deep convolutional neural network," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 2, pp. 257–269, Feb. 2018.
[5] Z. Xu, Y. Liu, and C. Li, "Distributed semi-supervised learning with missing data," *IEEE Trans. Cybern.*, vol. 51, no. 12, pp. 6165–6178, Dec. 2021.
[6] J. Zhu, Z. Ge, Z. Song, and F. Gao, "Review and big data perspectives on robust data mining approaches for industrial process modeling with outliers and missing data," *Annu. Rev. Control*, vol. 46, pp. 107–133, Oct. 2018.
[7] E. Hallaji, R. Razavi-Far, and M. Saif, "DLIN: Deep ladder imputation network," *IEEE Trans. Cybern.*, early access, Mar. 4, 2021, doi: 10.1109/TCYB.2021.3054878.
[8] 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, 2010.
[9] N. Marchang and R. Tripathi, "KNN-ST: Exploiting spatio-temporal correlation for missing data inference in environmental crowd sensing," *IEEE Sensor J.*, vol. 21, no. 3, pp. 3429–3436, Feb. 2021.
[10] L. Folguera, J. Zupan, D. Cicerone, and J. F. Magallanes, "Self-organizing maps for imputation of missing data in incomplete data matrices," *Chemometr. Intell. Lab. Syst*, vol. 143, pp. 146–151, Apr. 2015.
[11] G. Wang, J. Lu, K. Choi, and G. Zhang, "A transfer-based additive LS-SVM classifier for handling missing data," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 739–752, Feb. 2020.
[12] Y. Wang, Z. Pan, X. Yuan, C. Yang, and W. Gui, "A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network," *ISA Trans.*, vol. 96, pp. 457–467, Jan. 2020.
[13] H. Chen, B. Jiang, S. X. Ding, and B. Huang, "Data-driven fault diagnosis for traction systems in high-speed trains: A survey, challenges, and perspectives," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 1700–1716, Mar. 2022.
[14] K. Liu, Y. Tang, W. Lou, Y. Liu, J. Yang, and Y. Yao, "A thermographic data augmentation and signal separation method for defect detection," *Meas. Sci. Technol.*, vol. 32, no. 4, 2021, Art. no. 45401.
[15] A. W. Mulyadi, E. Jun, and H.-I. Suk, "Uncertainty-aware variational-recurrent imputation network for clinical time series," *IEEE Trans. Cybern.*, early access, Mar. 4, 2021, doi: 10.1109/TCYB.2021.3053599.
[16] Q. Ma, W. Lee, T. Fu, Y. Gu, and G. Yu, "MIDIA: Exploring denoising autoencoders for missing data imputation," *Data Min. Knowl. Disc.*, vol. 34, no. 6, pp. 1859–1897, 2020.
[17] J. Yoon, J. Jordon, and M. van der Schaar, "GAIN: Missing data imputation using generative adversarial nets," 2018, *arxiv:1806.02920*.
[18] V. Fortuin, D. Baranchuk, G. Rätsch, and S. Mandt, "GP-VAE: Deep probabilistic time series imputation," 2019, *arxiv:1907.04155*.
[19] I. Spinelli, S. Scardapane, M. Scarpiniti, and A. Uncini, "Efficient data augmentation using graph imputation neural networks," in *Progresses in Artificial Intelligence and Neural Systems*. Singapore: Springer, 2021, pp. 57–66.
[20] X. Lai, X. Wu, and L. Zhang, "Autoencoder-based multi-task learning for imputation and classification of incomplete data," *Appl. Soft Comput.*, vol. 98, Jan. 2021, Art. no. 106838.
[21] X. Lai, X. Wu, L. Zhang, W. Lu, and C. Zhong, "Imputations of missing values using a tracking-removed autoencoder trained with incomplete data," *Neurocomputing*, vol. 366, pp. 54–65, Nov. 2019.
[22] A. Kazemi and H. Meidani, "IGANI: Iterative generative adversarial networks for imputation with application to traffic data," 2020, *arxiv:2008.04847*.
[23] C. Shang, A. Palmer, J. Sun, K.-S. Chen, J. Lu, and J. Bi, "VIGAN: Missing view imputation with generative adversarial networks," 2017, *arxiv:1708.06724*.
[24] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 421–436.
[25] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," 2014, *arxiv:1412.6980*.
[26] T. Tieleman, and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.
[27] D. Xu, P. J. Hu, T. Huang, X. Fang, and C. Hsu, "A deep learning-based, unsupervised method to impute missing values in electronic health records for improved patient management," *J. Biomed. Inform.*, vol. 111, Nov. 2020, Art. no. 103576.
[28] L. Tran, X. Liu, J. Zhou, and R. Jin, "Missing modalities imputation via cascaded residual autoencoder," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1405–1414.
[29] Z. Zhou, J. Mo, and Y. Shi, "Data imputation and dimensionality reduction using deep learning in industrial data," in *Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, 2017, pp. 2329–2333.
[30] L. H. Chiang, E. L. Russell, and R. D. Braatz, "Fault detection and diagnosis in industrial systems," in *Advanced Textbooks in Control and Signal Processing*. London, U.K.: Springer, 2000.
[31] S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang, "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process," *J. Process Control*, vol. 22, no. 9, pp. 1567–1581, 2012.
[32] Q. Jiang, B. Huang, and X. Yan, "GMM and optimal principal components-based Bayesian method for multimode fault diagnosis," *Comput. Chem. Eng.*, vol. 84, pp. 338–349, Jan. 2016.
[33] N. Abiri, B. Linse, P. Edén, and M. Ohlsson, "Establishing strong imputation performance of a denoising autoencoder in a wide range of missing data problems," *Neurocomputing*, vol. 365, pp. 137–146, Nov. 2019.
[34] Z. Pan, Y. Wang, X. Yuan, C. Yang, and W. Gui, "A classification-driven neuron-grouped SAE for feature representation and its application to fault classification in chemical processes," *Knowl. Based Syst.*, vol. 230, Oct. 2021, Art. no. 107350.

Zhuofu Pan received the B.Eng. degree from the School of Civil Engineering, Changsha University of Science and Technology, Changsha, China, in 2014, and the M.Eng. degree from Central South University, Changsha, in 2017, where he is currently pursuing the Ph.D. degree with the School of Automation.

He is currently a visiting student with the Department of Chemical and Materials Engineering, University of Alberta, Edmonton, AB, Canada. His research interests include interpretable deep learning, process monitoring and diagnosis, fault attribution and reconstruction, data-based control theory, and neural network-simulated system dynamics.

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

Since 2003, she has been with the School of Information Science and Engineering, Central South University, where she was at first an Associate Professor and is currently a Professor. Her research interests include the modeling, optimization, and control for complex industrial processes, intelligent control, and process simulation.



Kai Wang (Member, IEEE) received the B.Eng. and Ph.D. degrees 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 with the School of Automation, Central South University, Changsha, China. He specializes in industrial data analytics, process health management, and machine learning.

Hongtian Chen (Member, IEEE) received the B.S. and M.S. degrees in electrical and automation engineering from Nanjing Normal University, Nanjing, China, in 2012 and 2015, respectively, and the Ph.D. degree in automation engineering from the Nanjing University of Aeronautics and Astronautics, Nanjing, in 2019.

He was a Visiting Scholar with the Institute for Automatic Control and Complex Systems, University of Duisburg–Essen, Duisburg, Germany, in 2018. He is currently a Postdoctoral Fellow with the Department of Chemical and Materials Engineering, University of Alberta, Edmonton, AB, Canada. His research interests include process monitoring and fault diagnosis, data mining and analytics, machine learning, and quantum computation; and their applications in high-speed trains, new energy systems, and industrial processes.

Chunhua Yang (Fellow, IEEE) received the M.Eng. degree in automatic control engineering and the Ph.D. degree in control science and engineering from 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 with Central South University. Her current research interests include modeling and optimal control of complex industrial process, intelligent control system, and fault-tolerant computing of real-time systems.

Weihua Gui received the B.Eng. and M.Eng. degrees from the Department of Control Science and Engineering, Central South University, Changsha, China, in 1976 and 1981, respectively.

From 1986 to 1988, he was a Visiting Scholar with Universität-GH-Duisburg, Duisburg, Germany. He has been a Full Professor with the School of Information Science and Engineering, Central South University since 1991. His main research interests include modeling and optimal control of complex industrial processes, distributed robust control, and fault diagnoses.

Prof. Gui was elected as an Academician of Chinese Academy of Engineering in 2013.