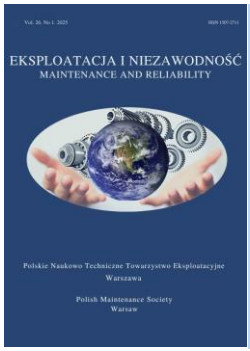


Eksploatacja i Niezawodnosc – Maintenance and Reliability

Volume 27 (2025), Issue 1

journal homepage: http://www.ein.org.pl

Article citation info:
Li X, Song K, Shi J, Degradation generation and prediction based on machine learning methods: A comparative study, Eksploatacja i Niezawodnosc – Maintenance and Reliability 2025: 27(1) http://doi.org/10.17531/ein/192168

Degradation generation and prediction based on machine learning methods: A comparative study

Indexed by: Web of Science Group

Xintong Li[a,b], Kai Song[a], Jian Shi[a,b,*]

[a] Academy of Mathematics and Systems Science, Chinese Academy of Sciences, China
[b] School of Mathematical Sciences, University of Chinese Academy of Sciences, China

Highlights

- The degradation trajectories are generated based on TimeGAN, SVAE, diffusion model, and the segmented sampling method, respectively.
- The diffusion model is used to synthesize the degradation process.
- Based on several generative models and predictive networks, evaluation and comparison of both numerical simulations and case studies are realized.

Abstract

In engineering practice, degradation analysis often suffers from the small-sample problem. To this end, several generative models are developed to expand the degradation data, based on which both the original data and the synthetic data are used to train neural networks for degradation prediction. However, these methods are rarely compared with each other, and the performances of competitive candidates are not explored. Given this, this paper reviews some machine learning-based methods and performs a comparison among them. Particularly, a segmented sampling method is proposed and the diffusion model is introduced for degradation generation. Results of both numerical simulations and case studies show that none of these methods can perform best in all cases, yet making use of synthetic data improves the predictive performance. Overall, the time-series generative adversarial network and the segmented sampling method are recommended for degradation generation, and the gated recurrent unit network is recommended for prediction.

Keywords
bootstrap, diffusion model, degradation analysis, generative models, degradation prediction.

1. Introduction

In modern society, many products are highly reliable so that there are few or no failures even after a long period of time. In this circumstance, performing reliability assessment based on lifetime data is hard to implement, and an alternative way is to utilize degradation signals because many failures can be attributed to an underlying degradation process. Usually, a failure occurs when the degradation level passes a threshold, which builds a natural link between degradation and lifetime [1]. Therefore, degradation analysis has gained great popularity in industry and academia.

However, owing to high costs, there are only a few units in degradation tests. Thus, degradation analysis is often confronted with the small-sample problem, and readers can see the degradation data sets reported in Meeker and Escobar [2] for some examples. Consequently, to address the issue of imprecise predictions attributable to the insufficient samples, it is imperative to augment the degradation dataset through the application of data generation technologies. In the case of small sample size, some researchers employed the generalized pivotal quantity (GPQ) method [3] to make inferences for degradation

(*) Corresponding author.
E-mail addresses: X. Li (ORCID: 0009-0001-1846-7771) lxt_2000@126.com, K. Song (ORCID: 0009-0008-6667-3431) songkaisubmit@126.com, J. Shi (ORCID: 0000-0002-3604-545X) jshi@iss.ac.cn,

models and found that this method usually outperformed the large-sample approximation method. Some of the studies on degradation analysis based on the GPQ method include Chen et al.[4], Wang et al. [5], Song and Cui [6], Chen and Ye [7], among others.

In literature, another way to tackle the small-sample challenge is to make use of generative models (GMs). Technically, GMs learn the underlying data distribution and then generate synthetic samples to enlarge the sample size. When dealing with insufficient data, the approach of generating synthetic data through learning the distribution of the small set of real data, as realistically as possible, has been widespread used across many domains. Lu et al. [8] summarized a thorough review of machine-learning (ML) method for synthetic data generation, and De et al. [9] presented a survey of GMs and their applications in the industrial internet of things. The generative adversarial network (GAN) [10] and the variational autoencoder (VAE) [11] are two of the most representative and widely used GMs. For example, Liu et al. [12] employed the GAN to obtain the synthetic fault data when the real fault samples are limited, and then developed a small-sample fault detection method for wind turbines. In the field of network intrusion detection, Liu et al. [13] proposed a VAE-based data augmentation scheme to address the problem of data imbalance. In addition, Booyse et al. [14] discussed the construction of deep digital twins based on the VAE and the GAN. Within the data mining sector, Yu et al. [15] introduced a novel data augmentation method that effectively enhances small datasets for clustering by capturing the underlying distribution of data with Gaussian Mixture Model-Wasserstein Generation Adversarial Network (GMM-WGAN). In the realm of agriculture, Zhang et al. [16] proposed generate adversarial-driven cross-aware network (GACNet) to improve the discriminatory accuracy of wheat variety identification. Zhang et al. [17] presented a Cascaded Visual Attention Network (CVANet) for single image super-resolution, leveraged a novel combination of feature, channel, and pixel attention modules to enhance detail reconstruction and outperform existing methods by better utilizing the complementary strengths of different feature representation layers. Recently, pertaining to the field of degradation, Shangguan et al. [18] employed the time-series GAN (TimeGAN), which was first proposed by Yoon et al. [19], to generate the synthetic wheel degradation data. What is more, due to the monotonicity of wheel degradation, they drew random vectors from a gamma process as input to generate synthetic data. Subsequently, Shangguan et al. [20] utilized the stacked VAE (SVAE), which connected several gated recurrent units (GRU)-based VAEs in series, for degradation generation, and they showed that the SVAE had a better performance than the traditional GAN and VAE by analyzing three real data sets. However, a direct comparison of TimeGAN and SVAE in synthesizing degradation data is not explored.

As a popular GM, the diffusion model [21] has been used to generate synthetic data in quite diverse domains. For instance, Ke et al. [22] addressed the problem of hard-constrained text generation via a meta-diffusion model. Zeng et al. [23] proposed an improved diffusion model by incorporating an autoencoder to detect radio anomalies. Especially, Dhariwal and Nichol [24] stated that diffusion models beat GANs on image synthesis. Gupta P. et al. [25] introduced a framework that utilizes diffusion models in few-shot image generation by modelling the latent space distribution, enhancing the diversity and fidelity of generated images despite limited sample sizes. Wang and Zhang [26] proposed a customized load profile synthesis method using conditional diffusion models tailored to heterogeneous customer data, improving data-driven model performance. Liu et al. [27] proposed a Perturbation-Assisted Sample Synthesis (PASS), generating synthetic data that accurately reflects the distribution of complex datasets, such as gene expression, images, and text, using perturbation techniques and pre-trained generative models, facilitating reliable statistical inference across various applications. Shen et al. [28] presented a Synthetic Data Generation for Analytics framework that enhances statistical methods using high-quality synthetic data from GMs like tabular diffusion, which improves accuracy and ensures privacy by adhering to the differential privacy standard, effectively addressing data scarcity and boosting performance across various case studies. Nevertheless, to our knowledge, how the diffusion model performs on degradation generation has not been checked yet.

The bootstrap is a sampling-based method, and it is usually used for assessing statistical accuracy [29]. For example, Zhou and Xu [30] constructed confidence intervals for model parameters by a naive bootstrap method which sampled the

whole measurements of each unit from the original data set with replacement. In Guo et al. [31], the bootstrap was used to generate degradation increments in a novel way. More specifically, when degradation increments were independent and identically distributed (I.I.D.), they sampled from all previous degradation increments to generate a realization of degradation increment in the next time interval. When the I.I.D. assumption was not satisfied, they first built parametric models for degradation increments and then combined the bootstrap method to generate future degradation increments. Unlike their methods, we apply the bootstrap to degradation increments of different units in the same time interval, and draw a sample with replacement for each disjoint time interval. By calculating the cumulative sum of these sampled degradation increments, we can obtain a new degradation path. We call this method the segmented sampling method in this paper. Clearly, the segmented sampling method is non-parametric, and can be used as a GM.

On the other hand, accurate prediction of future degradation levels is crucial to subsequent analyses, such as reliability evaluation, remaining useful life (RUL) estimation and condition-based maintenance. In view of the time-series characteristics of degradation data, it is appropriate to predict future degradation levels via recurrent neural networks, of which two important and commonly used members are the long-short term memory (LSTM) network [32] and the GRU network [33]. For example, Yang et al. [34] first constructed a composite health index by fusing multi-sensor data, and then applied the LSTM network to this health index for RUL prediction. Ungurean et al. [35] employed the GRU network to perform online predictions of battery health. Applications of the LSTM and GRU networks in sequence prediction can also be found in Lin et al. [36], Sun et al. [37] and Gu et al. [38], to name a few. Apart from these two classical networks, Transformer model [39] is also used for time-series prediction[40,41]. Unlike its predecessors that relied on recurrent or convolutional neural networks, the Transformer model adopts a purely attention-based mechanism, providing significant improvements in training efficiency and model performance across a range of sequence-to-sequence tasks. Notably, in previous studies [18,20], the GRU network was employed for degradation prediction, nevertheless, comparisons among with the LSTM network and Transformer model were not performed.

In summary, although a few deep learning-based generative methods have been employed to synthesize and predict degradation data, the performances of these methods are not compared. Besides, there are competitive alternatives that are not considered for degradation generation and prediction. The status motivates us to conduct a comparative study in this paper, and the contributions are summarized as follows:

- A segmented sampling method is developed as a GM.
- The diffusion model and the LSTM network as well as Transformer model are introduced for degradation generation and prediction, respectively.
- Both numerical simulations and case studies are conducted to evaluate and compare these promising methods.

The rest of this paper is structured as follows. Section 2 describes the problem. Section 3 presents three GMs along with a segmented sampling method for degradation generation. Section 4 introduces two neural network models for degradation prediction. Section 5 performs numerical simulations to evaluate different methods regarding the performances of degradation generation and prediction. Section 6 provides three illustrative examples. Finally, Section 7 gives a summary of this paper.

2. Problem Statement

Let $\{X(t), t \geq 0\}$ be a random degradation process, and its realizations of n independent test units are observed. Moreover, these units are inspected at time points $t_1, t_2, \cdots, t_m$, where m is the number of measurements. Then, the observed data set is $\{x_{ij}, i = 1, 2, \cdots, n, j = 1, 2, \cdots, m\}$, where $x_{ij} \triangleq x_i(t_j)$ denotes the degradation level of the ith unit at time t_j. Further, we denote $\Delta x_{ij} = x_{ij} - x_{i(j-1)}$ and $\Delta t_j = t_j - t_{j-1}$, where $x_{i0} = 0$ and $t_0 = 0$.

However, in engineering practice, due to the limitations of test costs, safety issue, experimental technology and other factors, the sample size is usually small. Reliability evaluation and subsequent decision-making based on insufficient degradation data may deviate from actualities. One possible remedy is to generate synthetic data, and then the original data and the synthetic data are combined to train advanced models that will be used for further analyses. This paper aims at the degradation generation and prediction, and explores the

generative performances of the TimeGAN, SVAE, diffusion model and a segmented sampling method and the predictive performances of the LSTM, GRU and Transformer networks. A flowchart is displayed in Figure 1 to clearly show the degradation generation and prediction framework.

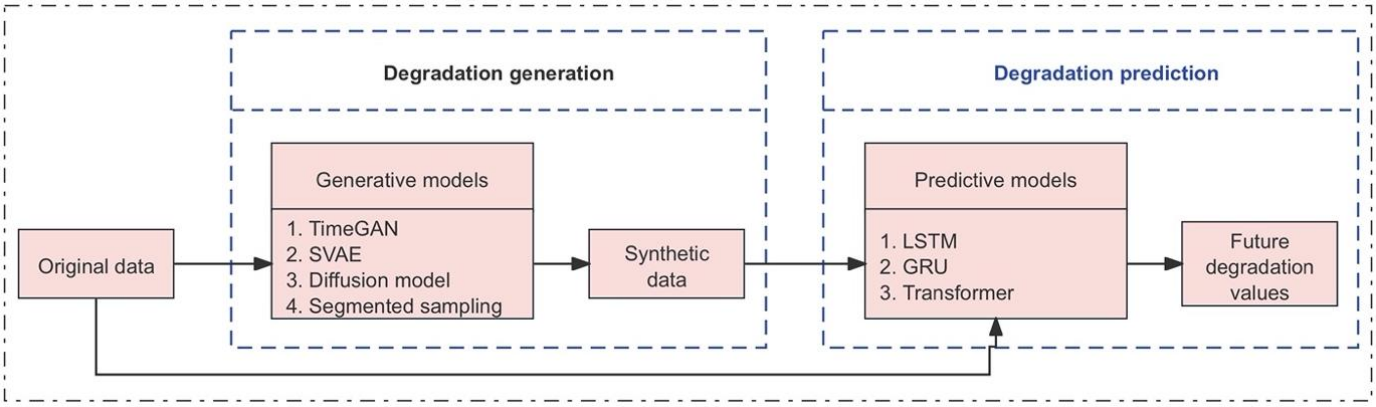


Figure 1. Framework of the degradation generation and prediction.

3. Degradation Generation

To begin with, we briefly summarize the TimeGAN and the SVAE, which were used for degradation generation in previous studies [18,20]. Then, we introduce a popular generative model, called the diffusion model, as a competitor in this paper. Besides the three deep learning generative methods, we also develop a segmented sampling method as a promising alternative.

3.1. TimeGAN

Besides the generator and discriminator in the classical GAN, the TimeGAN [19] also introduces an embedding network and a recovery network. The embedding network builds a mapping from original features to lower-dimensional representations, and also provides a latent space within which the generator and discriminator operate. The recovery network makes use of latent representations to reconstruct associated original data.

Let $f_E(\cdot)$ and $f_R(\cdot)$ be the mapping functions corresponding to the embedding network and the recovery network respectively, then given the input $A_{1:C}$, their latent representations and reconstructions are given by

$$H_c = f_E(H_{c-1}, A_c) \text{ and } \check{A}_c = f_R(H_c),\ c = 1,2,\cdots,C, \quad (1)$$

where $A_{1:C}$ is the sequence that is obtained by scaling the original degradation data via the minimax normalization method. To guarantee that the reconstructions and associated original data are close, a reconstruction loss function is established by

$$\Lambda_1 = E_A\left[\sum_c \left\|\check{A}_c - A_c\right\|_2\right], \quad (2)$$

where $\|\cdot\|_2$ is the L_2 -norm.

The generator produces synthetic data using random noise as input, and the discriminator judges whether a sample is real or synthetic. Here, the generator and discriminator are implemented via a recurrent network and a bidirectional recurrent network respectively, where the corresponding generating function and discrimination function are denoted as $f_G(\cdot)$ and $f_D(\cdot)$. Then, the synthetic latent code $\widehat{H}_c$ corresponding to random noise Z_c and the classification $\tilde{Y}_c$ of a latent code $\widetilde{H}_c$ are given by

$$\widehat{H}_c = f_G\left(\widehat{H}_{c-1}, Z_c\right) \text{ and } \tilde{Y}_c = f_D(\xi_c^B, \xi_c^F), \quad (3)$$

where $\xi_c^F = \chi^F\left(\widetilde{H}_c, \xi_{c-1}^F\right)$ and $\xi_c^B = \chi^B\left(\widetilde{H}_c, \xi_{c+1}^B\right)$ with $\chi^F(\cdot)$ and $\chi^B(\cdot)$ being the forward and backward recurrent functions respectively. Here, $\tilde{Y}_c \triangleq Y_c$ if $\widetilde{H}_c = H_c$, and $\tilde{Y}_c \triangleq \hat{Y}_c$ if $\widetilde{H}_c = \widehat{H}_c$ In reference [18], because of the monotonicity of degradation paths, Z_c was assumed to be gamma distributed. The generator and discriminator are updated alternatively until the Nash Equilibrium is achieved, and the corresponding unsupervised loss function can be written as

$$\Lambda_2 = E_A\left[\sum_c log(Y_c)\right] + E_Z\left[\sum_c log\left(1 - \hat{Y}_c\right)\right]. \quad (4)$$

To learn the temporal dynamics of sequential data more efficiently, a supervised loss function is introduced, that is,

$$\Lambda_3 = E_A\left[\sum_c \|H_c - f_G(H_{c-1}, Z_c)\|_2\right], \quad (5)$$

which measures the distance between the actual latent code and synthetic latent code produced by the generator.

Finally, the training of TimeGAN is performed by solving the following optimization problems.

$$\min_{\vartheta_E,\vartheta_R}(\delta_1\Lambda_3 + \Lambda_1) \text{ and } \min_{\vartheta_G}\left(\delta_2\Lambda_3 + \max_{\vartheta_D}\Lambda_2\right), \quad (6)$$

where δ_1 and δ_2 are hyperparameters, and $\vartheta_E, \vartheta_R, \vartheta_G$ and ϑ_D are unknown parameters corresponding to the embedding network, recovery network, generator and discriminator respectively. For illustration, in Figure 2, a pictorial description of the TimeGAN is presented.

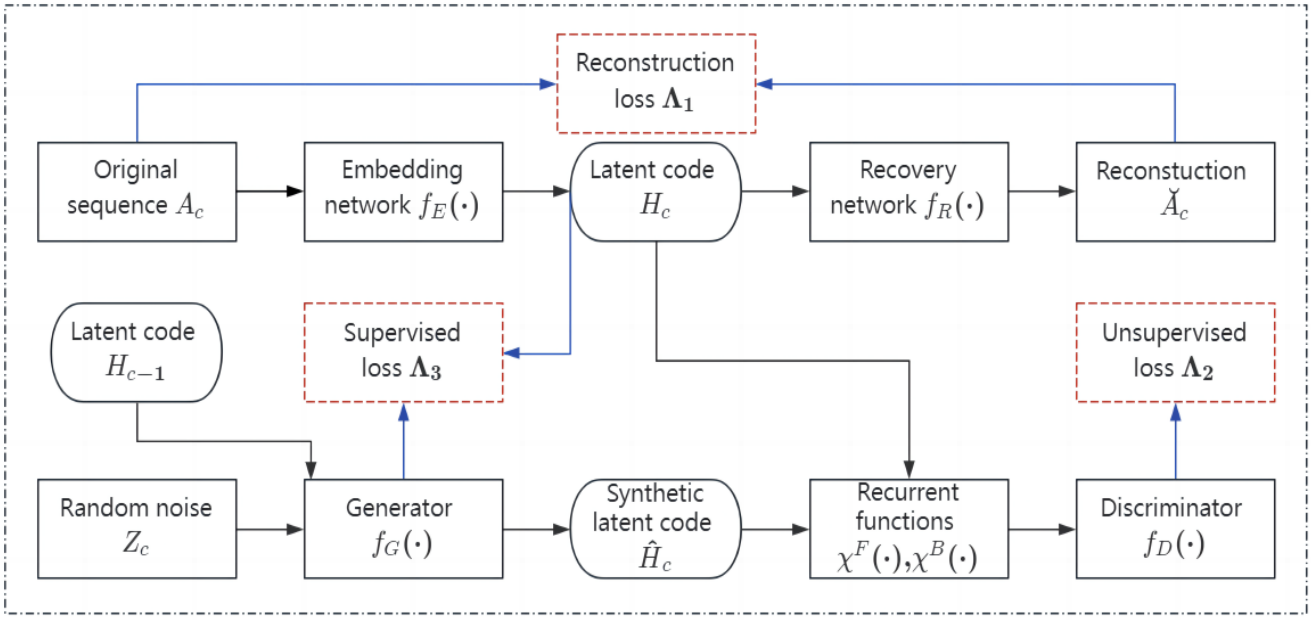


Figure 2. Framework of the TimeGAN.

3.2. SVAE

In reference [20], the VAE was extended by introducing the GRU network as a backbone firstly, then multiple GRU-based VAEs were stacked in series. The output of the current GRU-based VAE was the input of the next GRU-based VAE. The resulting model is called the SVAE for simplicity in this paper. Let Δ_c^s and $\hat{\Delta}_c^s$ be the outputs of the encoder and decoder corresponding to the sth GRU-based VAE respectively, where $s = 1,2,\cdots,S$ with S being the number of GRU-based VAEs stacked in the SVAE. Further, we denote $\hat{\Delta}_c^0 = A_c$, where A_c is the cth component of the input $A_{1:C}$.

In the s th GRU-based VAE, the modules of VAE are conditional on the hidden state h_{c-1}^s of the GRU network, which aims at capturing the temporal characteristics more efficiently. Specifically, the prior distribution of the latent variable Δ_c^s is given by

$$\Delta_c^s \sim N\left(\mu_{s,c}, \text{diag}\left(\sigma_{s,c}^2\right)\right) \text{ and } \left[\mu_{s,c}, \sigma_{s,c}^2\right] = \Psi_s^{prior}(h_{c-1}^s), \quad (7)$$

where $N(\varpi_1, \varpi_2)$ denotes the normal distribution with mean vector ϖ_1 and covariance matrix ϖ_2, and $\text{diag}\left(\sigma_{s,c}^2\right)$ denotes the diagonal matrix with the diagonal elements being $\sigma_{s,c}^2$. Besides, the generating distribution conditional on

$$\hat{\Delta}_c^s \triangleq \hat{\Delta}_c^{s-1}|\Delta_c^s \sim N\left(\breve{\mu}_{s,c}, \text{diag}\left(\breve{\sigma}_{s,c}^2\right)\right) \text{ and } \left[\breve{\mu}_{s,c}, \breve{\sigma}_{s,c}^2\right] = \Psi_s^{dec}\left(\psi_{1,s}(\Delta_c^s), h_{c-1}^s\right). \quad (8)$$

Finally, the approximate posterior of the latent variable Δ_c^s is

$$\Delta_c^s|\hat{\Delta}_c^{s-1} \sim N\left(\bar{\mu}_{s,c}, \text{diag}\left(\bar{\sigma}_{s,c}^2\right)\right) \text{ and } \left[\bar{\mu}_{s,c}, \bar{\sigma}_{s,c}^2\right] = \Psi_s^{enc}\left(\psi_{2,s}\left(\hat{\Delta}_c^{s-1}\right), h_{c-1}^s\right). \quad (9)$$

In Eqs. (7), (8) and (9), $\Psi_s^{prior}(\cdot)$, $\Psi_s^{dec}(\cdot)$, $\Psi_s^{enc}(\cdot)$, $\psi_{1,s}(\cdot)$ and $\psi_{2,s}(\cdot)$ are mapping functions that can be modelled by neural networks. What is more, the GRU network updates its hidden state by

$$h_c^s = f_s^{GRU}\left(\psi_{1,s}(\Delta_c^s), \psi_{2,s}\left(\hat{\Delta}_c^{s-1}\right), h_{c-1}^s\right), \quad (10)$$

where $f_s^{GRU}(\cdot)$ is the transition function of the GRU network, and one can see Section 0 for more details.

Training is performed based on the variational inference method, and the resulting loss function can be written as

$$\Upsilon = \sum_s E_{Q(\Delta^s_{1:C}|\hat{\Delta}^{s-1}_{1:C})} \left[\sum_c \begin{pmatrix} -\mathrm{KL}\left(Q(\Delta^s_c|\hat{\Delta}^{s-1}_{1:c}, \Delta^s_{1:c-1}) \parallel P(\Delta^s_c|c, \Delta^s_{1:c-1})\right) + \\ log\left(P(\hat{\Delta}^{s-1}_{1:c}|\Delta^s_{1:c}, \hat{\Delta}^{s-1}_{1:c-1})\right) \end{pmatrix}\right], \quad (11)$$

where the distributions $P(\Delta^s_c|\hat{\Delta}^{s-1}_{1:c-1}, \Delta^s_{1:c-1})$, $P(\hat{\Delta}^{s-1}_c|\Delta^s_{1:c}, \hat{\Delta}^{s-1}_{1:c-1})$ and $Q(\Delta^s_c|\hat{\Delta}^{s-1}_{1:c}, \Delta^s_{1:c-1})$ are defined by Eqs. (7), (8) and (9) respectively, and $Q(\Delta^s_{1:C}|\hat{\Delta}^{s-1}_{1:C}) = \prod_c Q(\Delta^s_c|\hat{\Delta}^{s-1}_{1:c}, \Delta^s_{1:c-1})$. Here, $KL(\cdot\|\cdot)$ denotes the Kullback-Leibler divergence between two distributions. In the end, Figure 3 presents a graphical illustration of the SAVE.

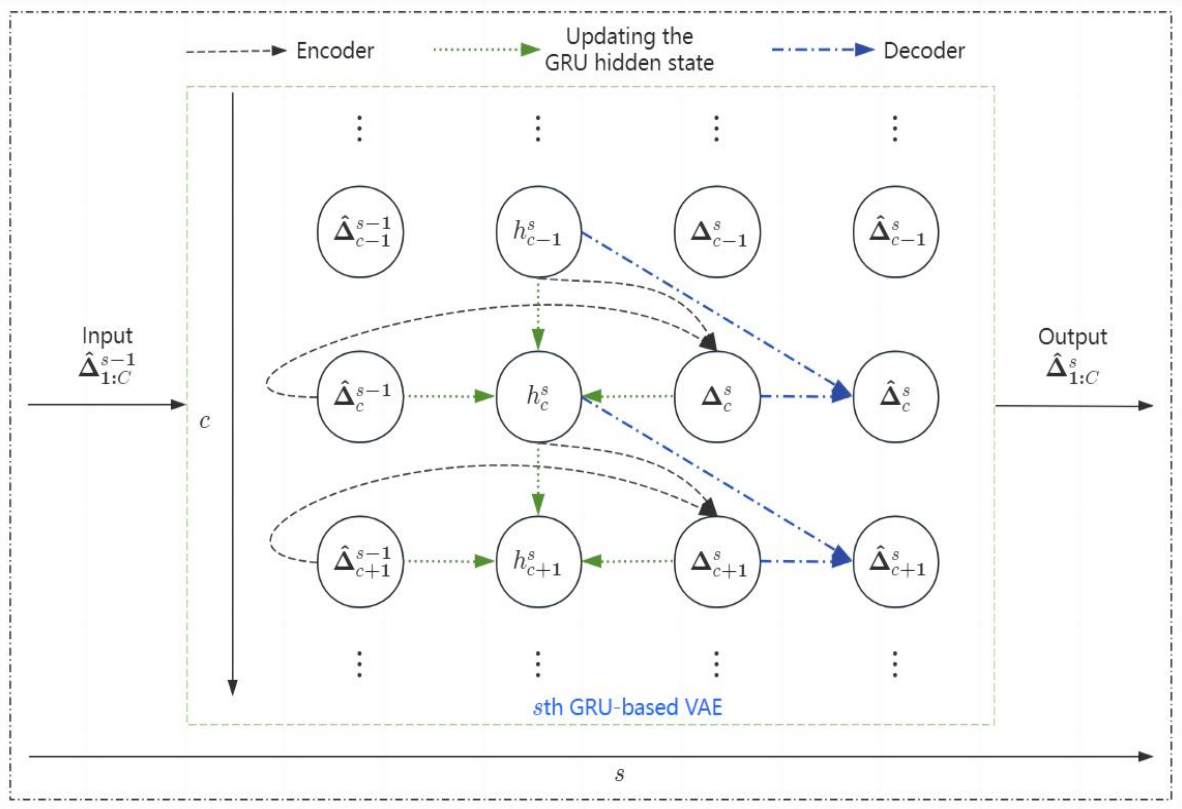


Figure 3. Framework of the SVAE.

3.3. Diffusion

The denoising diffusion probabilistic model [21], which is referred as to the diffusion model for brevity in this paper, consists of a forward process and a reverse process. More specifically, let V be the total number of diffusion steps, then the forward process adds noise to the data $\mathbf{U}_0$ step by step via the following equation

$$\mathbf{U}_v = \sqrt{1-\beta_v}\mathbf{U}_{v-1} + \sqrt{\beta_v}\epsilon_v,\ v = 1,2,\cdots,V, \quad (12)$$

where ϵ_v follows the standard Gaussian distribution, and β_v is a variance schedule that can be kept constant as a hyperparameter. This process terminates until $\mathbf{U}_V$ approximates the standard Gaussian noise.

The reverse process starts from standard Gaussian noise, and generates synthetic data by subtracting noises step by step via the following equation

$$\mathbf{U}_{v-1} = \mu_\theta(\mathbf{U}_v, v) + \sqrt{\gamma_v}\epsilon_v,\ v = V, V-1,\cdots,1, \quad (13)$$

where γ_v and $\mu_\theta(\mathbf{U}_v, v)$ are given by

$$\gamma_v = \beta_v \text{ or } \frac{1-\bar{\alpha}_{v-1}}{1-\bar{\alpha}_v}\beta_v \text{ and } \mu_\theta(\mathbf{U}_v, v) = \frac{1}{\sqrt{\alpha_v}}\left(\mathbf{U}_v - \frac{1-\alpha_v}{\sqrt{1-\bar{\alpha}_v}}\epsilon_\theta(\mathbf{U}_v, v)\right), \quad (14)$$

in which $\alpha_v = 1-\beta_v$ and $\bar{\alpha}_v = \prod_{l=0}^{v}\alpha_l$. Here, $\epsilon_\theta(\mathbf{U}_v, v)$ is usually a neural network model with parameter θ, which aims at predicting the noise from $\mathbf{U}_v$.

Training is done by minimizing the following objective function

$$L(\theta) = E_{v,\mathbf{U_0},\epsilon}\left[\left|\left|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_v}\mathbf{U}_0 + \sqrt{1-\bar{\alpha}_v}\epsilon, v\right)\right|\right|_2^2\right], \quad (15)$$

where v and ϵ follow the uniform distribution on $\{1,2,\cdots,V\}$ and the standard Gaussian distribution respectively. In the case of degradation generation, $\mathbf{U_0}$ is the vector which is composed of degradation increments corresponding to a degradation path, that is, $\mathbf{U_0} = [\Delta x_1, \Delta x_2, \cdots, \Delta x_m]$. In addition, a schematic diagram is given in Figure 4 to show how the diffusion model

works.

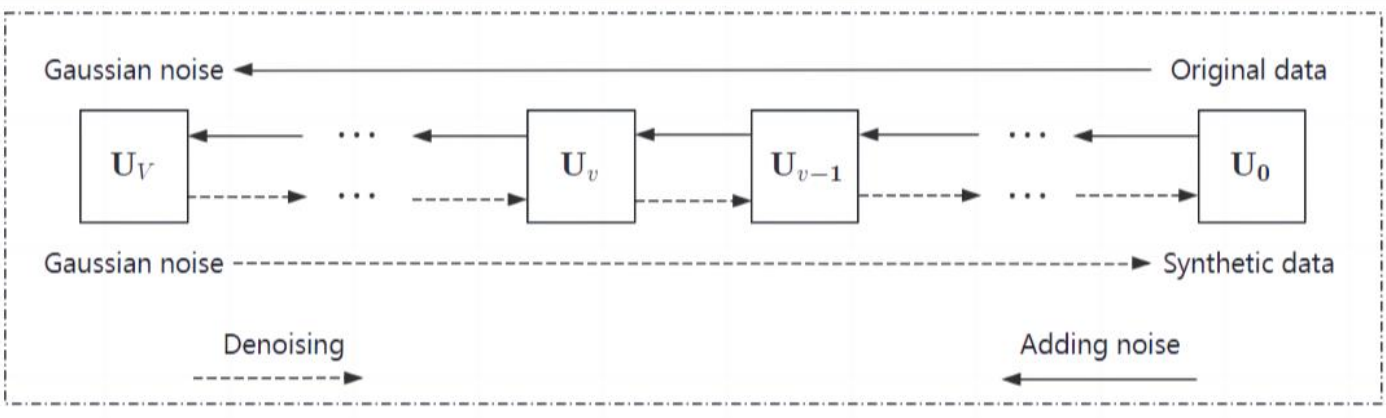


Figure 4. Framework of the diffusion model.

3.4. Segmented Sampling method

The basic idea of the segmented sampling method is to sample with replacement from degradation increments in each disjoint time interval independently. Then, the resampled degradation increments are summed cumulatively to form a new degradation path. The fact that degradation increments of different units in the same time interval are I.I.D. provides the theoretical foundation of this method.

The implementation procedures of the segmented sampling method are summarized as follows:

Step 1. For $j = 1,2,\cdots,m$, draw a sample Δx_j^* from the set $\{\Delta x_{1j}, \Delta x_{2j}, \cdots, \Delta x_{nj}\}$ uniformly;

Step 2. A new degradation path is given by $[x^*(t_1), x^*(t_2), \cdots, x^*(t_m)]$, where $x^*(t_j) = \sum_{l=1}^{j} \Delta x_j^*$;

Step 3. Repeat Step 1 and Step 2 n times to obtain a synthetic data set which has the same sample size and inspection times as the original data set.

4. Degradation Prediction

This section reviews the LSTM and GRU networks, which are two popular methods to analyse sequential data. For convenience, we use the time step k to denote time point t_k in this section.

4.1. LSTM

The LSTM network [32] replaces traditional nodes in the hidden layer via memory cells which employ input gates, forget gates and output gates to control the flow of information. When the current input information x_k at time step k and the previous hidden state h_{k-1} at time step $(k-1)$ are available, the input gate I_k, the forget gate F_k and the output gate O_k at time step k are calculated by

$$\begin{aligned} I_k &= \sigma(w_{ix}x_k + w_{ih}h_{k-1} + b_i), \\ F_k &= \sigma\left(w_{fx}x_k + w_{fh}h_{k-1} + b_f\right) \text{ and} \\ O_k &= \sigma(w_{ox}x_k + w_{oh}h_{k-1} + b_o), \end{aligned} \tag{16}$$

where $\sigma(\cdot)$ is the sigmoid activation function. Meanwhile, a candidate of memory cell internal state $\tilde{C}_k$ can be deduced by

$$\tilde{C}_k = tanh(w_{cx}x_k + x_{ch}h_{k-1} + b_c), \tag{17}$$

where $tanh(\cdot)$ is the hyperbolic tangent activation function. In Eqs. (16) and (17), w_{ly} and b_l are the weight parameter and the bias parameter respectively, where $l \in \{i, f, o, c\}$ and $y \in \{x, h\}$. Then, the memory cell internal state C_k at time step k is updated by

$$C_k = F_k \odot C_{k-1} + I_k \odot \tilde{C}_k, \tag{18}$$

where $\odot$ is the point-wise product operator. From Eq. (18), we can see that the forget gate F_k controls how much information is retained from the old memory cell internal state C_{k-1}, and the input gate I_k determines the amount of new input information that should be added to the current cell state. Finally, the hidden state h_k at time step k is given by

$$h_k = O_k \odot tanh(C_k), \tag{19}$$

where the output gate O_k governs the impact of the memory cell internal state C_k on subsequent layers. Further, the structure of the LSTM network is depicted in Figure 5 as a graphical illustration.

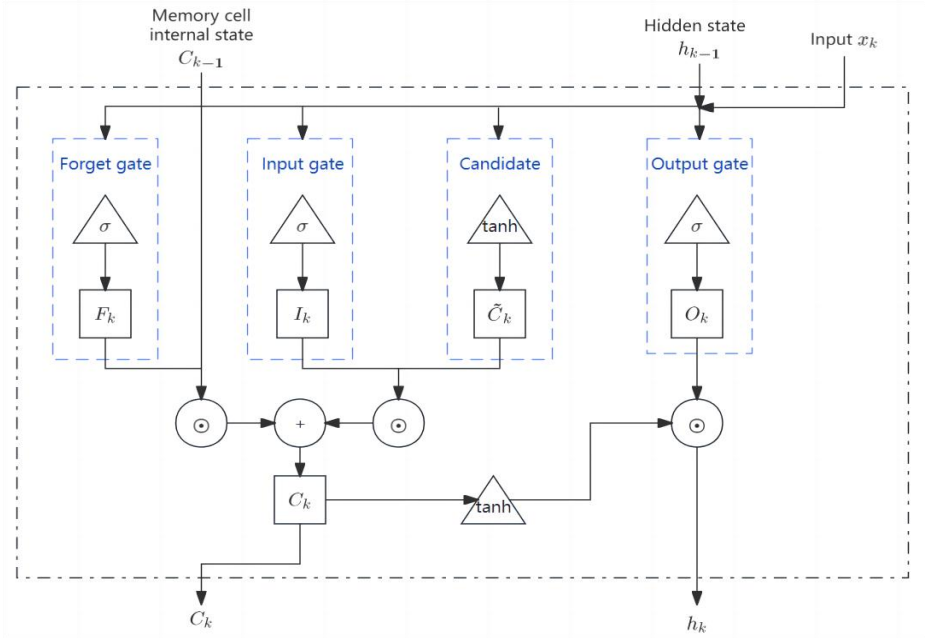


Figure 5. Structure of the LSTM network.

4.2. GRU

The GRU network [33] is also based on the multiplicative gating mechanisms, nevertheless, there are only two gates, that is, the reset gate and the update gate. Given the input x_k at the current time step and the hidden state h_{k-1} at the previous time step, the reset gate r_k and the update gate z_k at time step k are computed by

$$r_k = \sigma(w_{rx}x_k + w_{rh}h_{k-1} + b_r)$$
$$\text{and } z_k = \sigma(w_{zx}x_k + w_{zh}h_{k-1} + b_z), \qquad (20)$$

based on which a candidate hidden state $\tilde{h}_k$ at time step k is given by

$$\tilde{h}_k = tanh(w_{hx}x_k + w_{hh}(r_k \odot h_{k-1}) + b_h), \qquad (21)$$

where w_{qp} and b_q are respectively the weight parameter and the bias parameter with $q \in \{r, z, h\}$ and $p \in \{x, h\}$. From Eq. (21), we can see that the reset gate r_k controls the influence of the previous hidden state h_{k-1} on the candidate $\tilde{h}_k$. Finally, a weighted average of the old state h_{k-1} and the new candidate state $\tilde{h}_k$ is calculated to obtain the hidden state h_k at time step k, that is,

$$h_k = z_k \odot h_{k-1} + (1 - z_k) \odot \tilde{h}_k, \qquad (22)$$

where the update gate z_k serves as a weighting coefficient. Similarly, in Figure 6, we provide a diagram of the GRU structure.

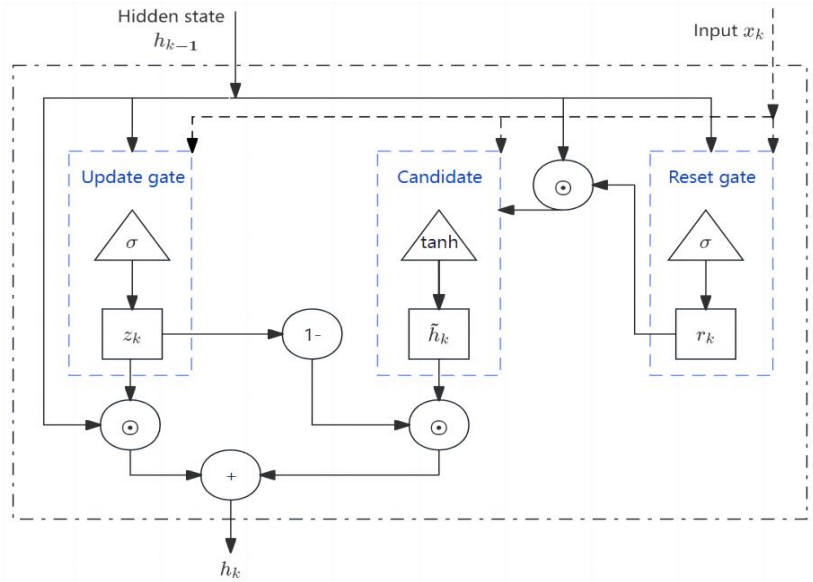


Figure 6. Structure of the GRU network.

4.3. Transformer

The Transformer model [39] is a good candidate for time series forecasting by taking advantage of the multi-head self-attention mechanism, since self-attention enables Transformer to capture both long- and short-term dependencies, and different attention heads learn to focus on different aspects of temporal patterns. Given the input x_k at the current time step and the hidden state h_{k-1} at the previous time step, the input is encoded by Positional Encoding layer. Since the transformer model does not inherently process sequence order (unlike RNNs), positional encodings provide the model with information about the relative or absolute position of the tokens in the sequence. Typically, these encodings are implemented using sine and cosine functions of different frequencies, that is,

$$\begin{cases} PE(pos, 2i) = sin\left(pos/10000^{2i/d_{model}}\right); \\ PE(pos, 2i+1) = cos\left(pos/10000^{2i/d_{model}}\right), \end{cases} \quad (24)$$

where pos means the position in x_k, that is $pos = 0, 1, \cdots, m$, and $i = 0,1, \cdots, d_{model}/2 - 1$. Then, the input x_k and corresponding position encode are added together as $\mathbf{Y}_k$. Multi-Head Attention, which is an extension of the attention mechanism that allows the model to jointly attend to information from different representation subspaces at different positions, comprises self-attention sublayers, transferring $\mathbf{Y}_k$ into H sets of matrices: query matrices $\mathbf{Q}_h = \mathbf{Y}_k\mathbf{W}_h^Q$, key matrices $\mathbf{K}_h = \mathbf{Y}_k\mathbf{W}_h^K$, and value matrices $\mathbf{V}_h = \mathbf{Y}_k\mathbf{W}_h^V$, with $h = 1, \cdots, H$. Here, $\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V$ are learnable parameters. Then, the sequence of vector output is given by

$$\mathbf{O}_h = \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h) = softmax\left(\frac{\mathbf{Q}_h\mathbf{K}_h^T}{\sqrt{d_k}} \cdot \mathbf{M}\right)\mathbf{V}_h. \quad (25)$$

As for the decoder, the hidden state h_{k-1} is multiplied with a mask matrix, which is applied to filter the rightward attention by setting upper triangular elements to $-\infty$ in order to avoid future information leakage, and then computes Eqs. (24), (25). Subsequently, the output of the decoder d_k is calculated by the following attention layer, with matrices $\mathbf{Q}$, $\mathbf{K}$ from encoder and $\mathbf{V}$ from decoder. Consequently, the hidden state h_k at time step k is updated by

$$h_k = softmax(w_d \cdot d_k + b_d), \quad (26)$$

where w_d and b_d are respectively the weight parameter and the bias parameter.

The framework of the Transformer model is presented in Figure 7. In the structure of encoder and decoder part, the multi-head attention is staked with the positional encoding, two layers of fully-connected network, and a ReLU activation.

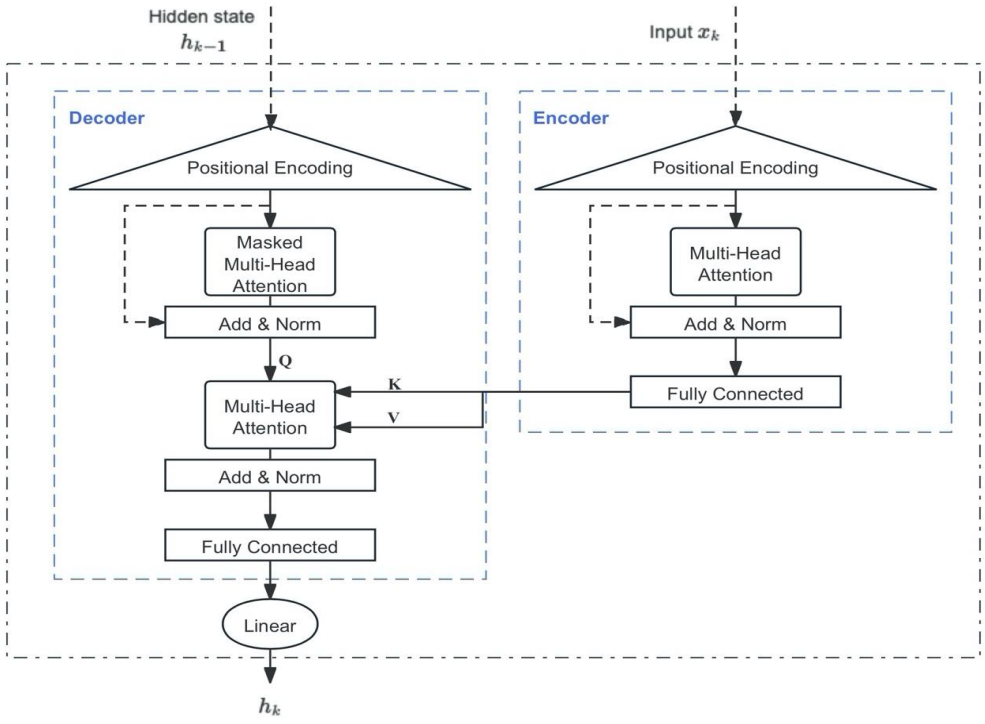


Figure 7. Structure of the Transformer model.

Note that there are some hyper-parameters in the above deep learning-based generative models and predictive networks, and they are specified in the appendix for the following numerical simulations and case studies.

5. Numerical Simulation

In this section, simulation studies are conducted to compare the generation and prediction performances of different methods.

5.1. One-dimensional Simulation

Here, the degradation data are generated using the inverse Gaussian (IG) process with mean function μt and shape parameter λ, denoted as $IG(\mu t, \lambda t^2)$. The true model parameters are set as $\mu = 0.1$ and $\lambda = 1$. Besides, we suppose that there are n test units, and each unit is inspected at time points $t_j = j * \Delta t, j = 0,1,2,\cdots,m$, where $\Delta t = 1$ is the inspection frequency. In the following text, we consider $(n,m) = (10,10), (10,20), (20,10)$ and $(20,20)$ for illustrative purposes. For each case, we use the first 80% of a simulated data set as the training set, and the remaining 20% of this data set forms the testing set to validate the generation performances of the GMs. After the training and evaluation process, four GMs generate degradation data which include n^* paths respectively, where $n^* = 0.8 * n$.

For the original data set, the degradation increments Δx_{ij}s are I.I.D. and follow the IG distribution $IG(0.1,1)$, where $i = 1,2,\cdots,n^*, j = 1,2,\cdots,m$ and $n^* = 0.8 * n$. Let $\Delta\tilde{x}_{ij}$ s be degradation increments corresponding to a synthetic data set, then we can evaluate the generative performance by comparing the distribution of $\Delta\tilde{x}_{ij}$ with $IG(0.1,1)$. Based on samples $\Delta\tilde{x}_{ij}$s, we can obtain the probability density function (PDF) of the synthetic increment using the kernel density estimation method. For each GM, one hundred synthetic data sets with the sample size n^* are generated, and the PDF of the synthetic increment is estimated for each synthetic data set, based on which an average PDF, denoted as $\bar{f}_{syn}(\Delta\tilde{x})$, can be calculated. Figure 8 shows the $\bar{f}_{syn}(\Delta\tilde{x})$s corresponding to different GMs, along with the PDF of $IG(0.1,1)$. It can be seen that the PDF curves of the four GMs are relatively close to the PDF curve of $IG(0.1,1)$. Intuitively, the extent of the closeness between the true PDF and the PDFs learned by the GMs is improved as the data size increases. These results indicate that the four GMs can learn the data distribution effectively. However, we also find that the SVAE and the diffusion model may produce negative degradation increments, which is in conflict with the fact that the IG-distributed increments are always positive. Further, the quality of synthetic data is assessed quantitatively. The Kolmogorov-Smirnov (K-S) statistic is defined as

$$D_{K-S} = \max_{\Delta\tilde{x}} \left| \hat{F}(\Delta\tilde{x}) - F_{IG}(\Delta\tilde{x}) \right|, \tag{27}$$

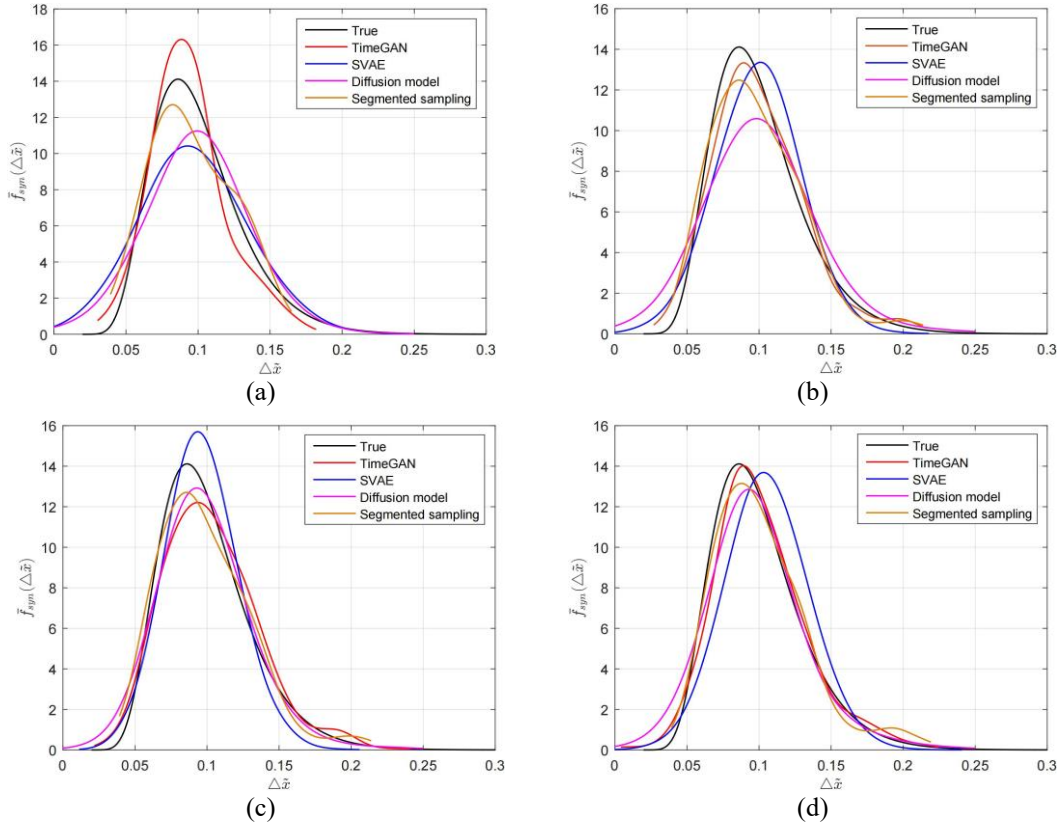


Figure 8. Comparison of the PDFs. (a) (n,m)=(10,10), (b) (n,m)=(10,20), (c) (n,m)=(20,10), (d) (n,m)=(20,20).

where $\hat{F}(\cdot)$ is empirical cumulative distribution function (CDF) calculated from synthetic increments $\Delta\tilde{x}_{ij}$s, and $F_{IG}(\cdot)$ is the CDF of $IG(0.1, 1)$. The K-S statistic measures the difference between the empirical CDF and the hypothesized CDF, so the smaller the better. The average D_{K-S} value over one hundred synthetic data sets is calculated under different (n, m)s for each GM, and the results are displayed in Table 1. As we can see, for each GM, the average D_{K-S} value decreases as n or m increases, suggesting that a larger data size usually leads to higher-quality synthetic data. Besides, the segmented sampling method has the smallest average D_{K-S} value under different (n, m) s, demonstrating its superiority. For the other three deep learning-based GMs, it appears that the TimeGAN and the diffusion model have some advantages.

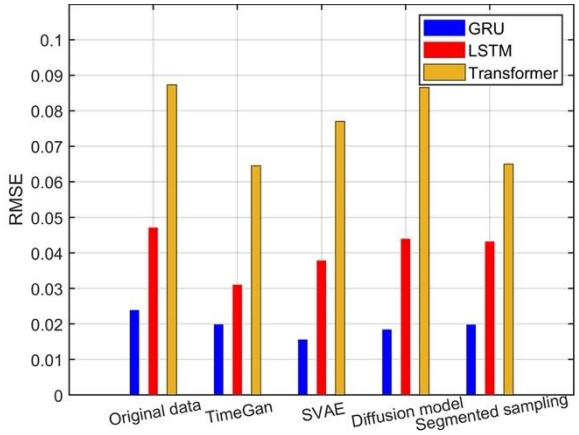
Table 1. Average D_{K-S} values of the four GMs under different (n, m).

(n, m)	TimeGAN	SVAE	Diffusion model	Segmented sampling
(10, 10)	0.1152	0.1260	0.1215	0.1098
(10, 20)	0.0942	0.0967	0.1062	0.0795
(20, 10)	0.0886	0.0992	0.0935	0.0836
(20, 20)	0.0666	0.0857	0.0624	0.0515

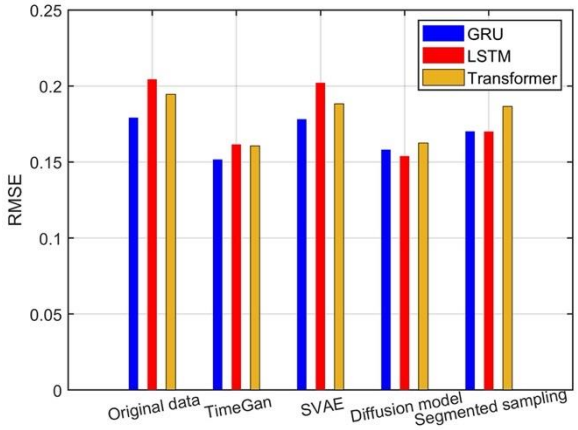
Regarding the degradation prediction, the original training set and five synthetic data sets are combined to train the LSTM, GRU and Transformer networks, which are then used to predict the degradation values of units in the testing set at the last 20% inspection time points. To quantify the predictive accuracy, the root mean squared error is adopted, which is defined as

$$\text{RMSE} = \left[\frac{1}{n-n^*}\sum_{i=n^*+1}^{n}(\hat{x}_{im} - x_{im})^2\right]^{\frac{1}{2}}, \qquad (28)$$

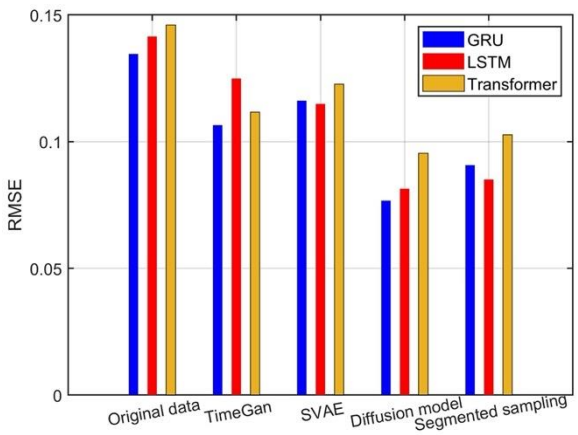
where $\hat{x}_{im}$ is the predictive value of x_{im}, at last 20% time points, through prediction networks (LSTM, GRU and Transformer). Note that each path of degradation data in original data or/and in synthetic data is the input of the predictive networks, whose first 80% time points is used to predict the counterpart of degradation data. The training and testing split remained consistent, using 80% of paths for training and 20% for testing, irrespective of whether the input data to the prediction network consisted of original data alone or a combination of original and synthetic data. And Figure 9 displays the RMSEs of the LSTM, GRU and Transformer networks when synthetic data sets are generated by different GMs. Here, for comparison, we also report the RMSEs corresponding to the case where only the original training set is used to train the predictive networks. It also embodies the concept of ablation studies, which substantiates the significance of generative models for predicting degenerated data through comparative validation. From Figure 9, we can see that incorporation of synthetic data into the original training set improves the predictive performance of the LSTM, GRU and Transformer networks no matter which GM is used. Besides, the predictive performance of the GRU network is comparable or superior to that of the LSTM network or the Transformer model in most cases. In addition, the Transformer model has relatively better performance when the length of time series data is longer, a characteristic that is also observed in the LSTM network.



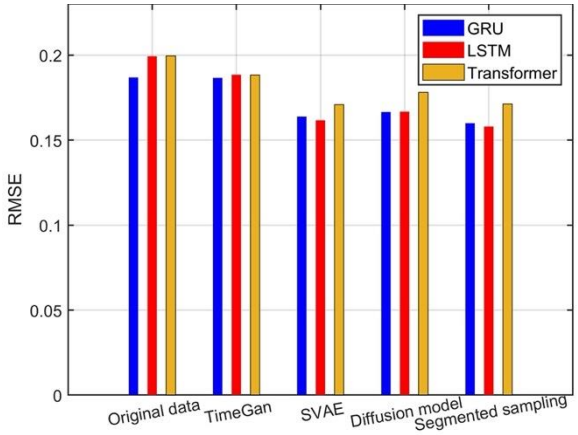
(a)



(b)



(c)



(d)

Figure 9. Results of the degradation prediction. (a) (n, m)=(10,10), (b) (n, m)=(10,20), (c) (n, m)=(20,10), (d) (n, m)=(20,20).

5.2. Multi-dimensional Simulation

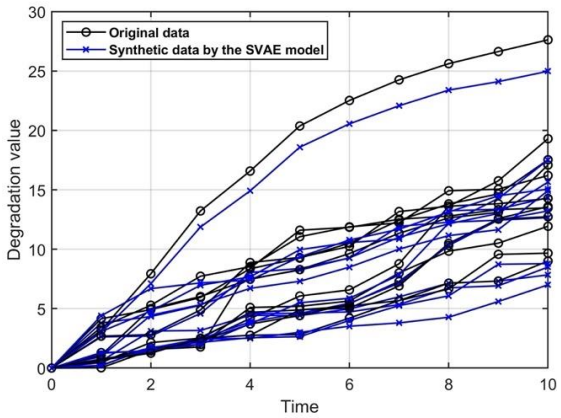
Here, the two-dimensional degradation data are generated using the bivariate degradation model proposed by Song and Cui [42]. Conditional on a common latent variable λ, this model uses two independent gamma processes to describe marginal degradation processes. Nevertheless, when variable λ varies according to a specific probability distribution, the two degradation processes are dependent, because λ has effects on the two degradation processes simultaneously. The bivariate degradation model is given by

$$X_1(t)|\lambda \sim Gamma\left(\lambda\mu_1, a_1\eta(t;b_1)\right) \text{ and } X_2(t)|\lambda \sim Gamma\left(\lambda\mu_2, a_2\eta(t;b_2)\right), \quad (29)$$

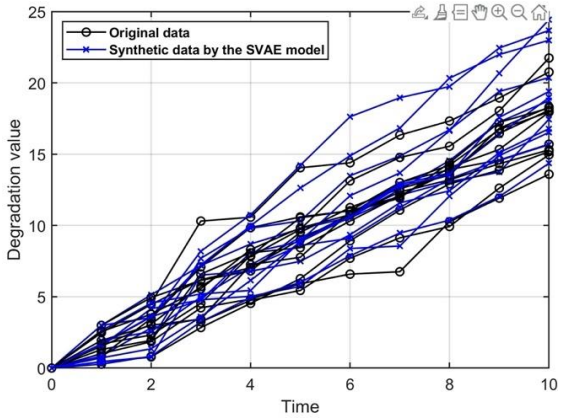
where $\eta(t;b_i) = t^{b_i}, i = 1,2$, and $\lambda \sim Gamma(\beta,\beta)$. Our true model parameters are set as, $(\mu_1, a_1, b_1, \mu_2, a_2, b_2, \beta) = (1,1,1,2,1.5,1,4)$. Besides, we suppose that there are n test units, and each unit is inspected at time points $t_j = j * \Delta t, j = 0,1,2,\cdots,m$, where $\Delta t = 1$ is the inspection frequency. The degradation data in each time point have two dimensions. In the following text, we consider $(n, m, dim) = (20, 10, 2)$ for illustrative purposes.

To illustrate, Figure 10 presents the original training datasets alongside synthetic data generated by the three GMs for the two-dimensional degradation dataset across each dimension. Note that, the TimeGAN model in our paper, developed for generating data from degenerated distributions, requires the addition of gamma noise. While this is straightforward in the unidimensional case, the feasibility of applying it in multidimensional scenarios necessitates a more thorough theoretical analysis. Therefore, this section does not explore the TimeGAN model in multidimensional contexts.

Figure 10 suggests that the degradation trend has been effectively captured, as the synthetic data largely align with the original datasets, demonstrating satisfactory model performance. Furthermore, to assess the generative capabilities of the three GMs more rigorously, the average D_{K-S} value over one hundred synthetic data sets is calculated under different dimensions for each GM. The results are summarized in Table 2. The segmented sampling method exhibits the lowest average D_{K-S} values across different dimensions, underscoring its superior performance.



(a)



(b)

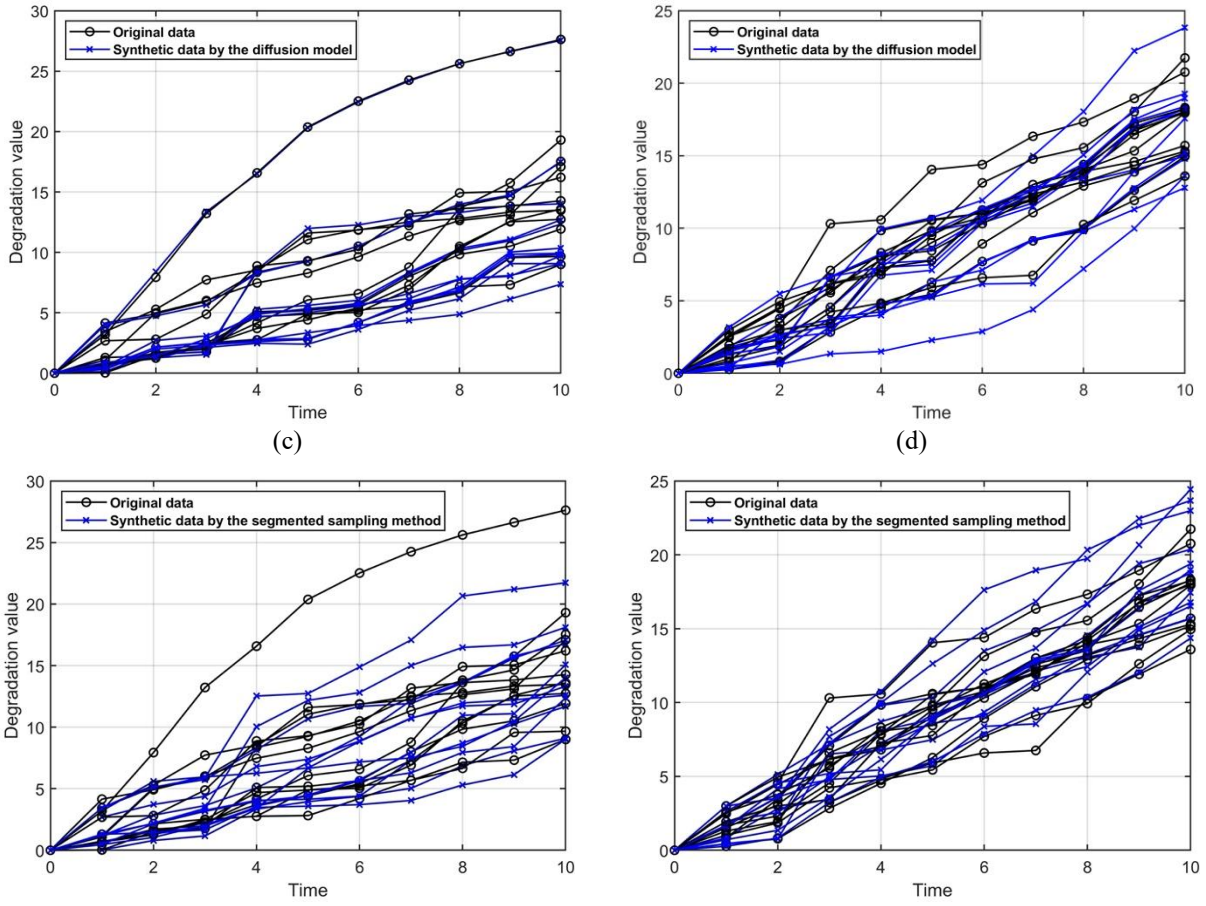


Figure 10. Illustration of the original data and the synthetic data for each dimension. (a)(c)(e) First dimension, (b)(d)(f) Second dimension.

Table 2. Average D_{K-S} values of the three GMs under each dimension.

d	TimeGAN	SVAE	Diffusion model	Segmented sampling
1	-	0.0831	0.0829	0.0825
2	-	0.0896	0.1067	0.0732

Concerning degradation prediction, the original training dataset is augmented with five synthetic datasets to train LSTM, GRU, and Transformer networks. These models are subsequently employed to forecast the two-dimensional degradation values for units in the test set. Figure 11 displays the RMSEs of the LSTM, GRU and Transformer networks when synthetic data sets are generated by different GMs. Here, the RMSEs are calculated by two dimensions of degradation prediction together. It is evident that the RMSE s are consistently lower when any of the synthetic datasets generated by the three GMs are utilized, compared to those achieved using only the original dataset. This observation substantiates the ability of the three GMs to capture the underlying characteristics of the original degradation data effectively.

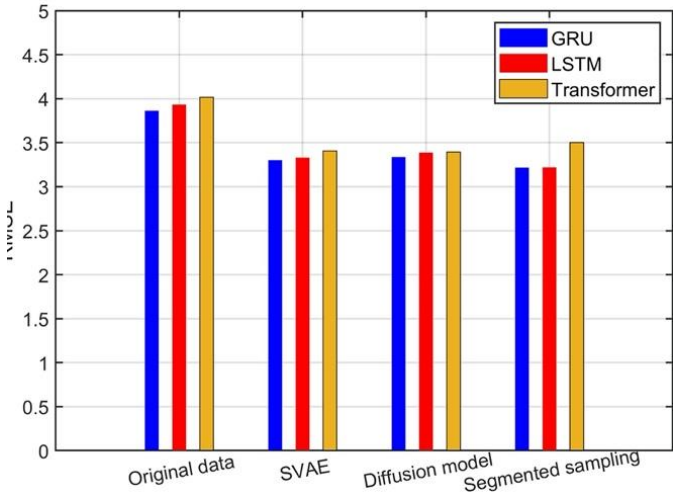


Figure 11. Results of the two-dimensional degradation prediction.

Most importantly, based on the two-dimensional results presented above, we have demonstrated that the degradation generation and prediction models are effective for two-dimensional degradation data, and by extension, are applicable to multi-dimensional degradation paths as well.

6. Case Study

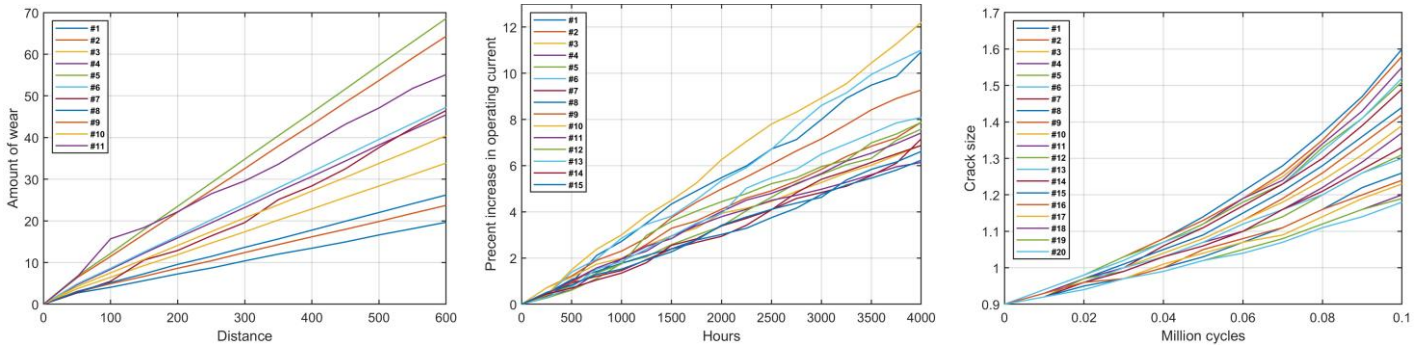
This section illustrates different degradation generation and prediction methods by analysing the following three data sets.

(1) Train wheel degradation data [43,44]. There are fourteen wheels, and they are inspected at $t_j = j * 50$ (distance in 10^3 km), where $j = 0,1,\cdots,12$. The degradation measurement is the amount of wear (in mm). Three units reach the failure threshold during the test time, and they are not considered in this study.

(2) GaAs laser degradation data [2]. Fifteen GaAs lasers are tested at 80°C, and their percent increases in operating current are measured with observation frequency of 250 hours and termination time of 4000 hours.

(3) Fatigue-Crack-Growth degradation data [2]. Fatigue-Crack-Growth data gives the size of fatigue cracks as a function of number of cycles of applied stress for 21 test specimens. These degradation data collect different length of cycles, and we consider 0.00-0.10 million cycles in this study.

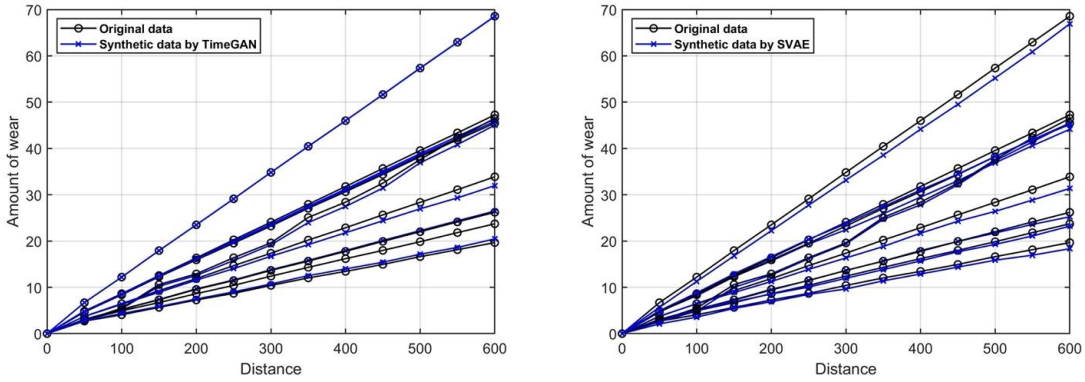
Figure. 12 displays the corresponding degradation paths of the three data sets. It is worth noting that the train wheel and GaAs laser data paths are approximately linear, whereas the fatigue crack degradation paths are non-linear, thus ensuring a comprehensive analysis in the case studies.



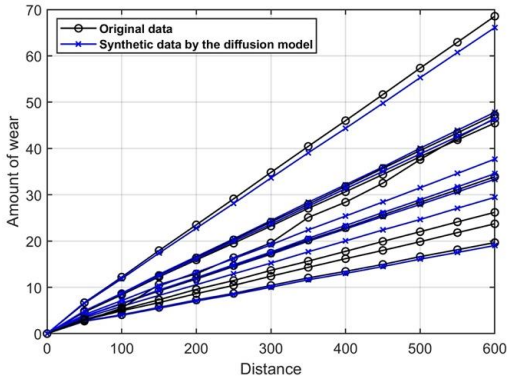
(a) (b) (c)

Figure. 12 Degradation paths. (a) Train wheel, (b) GaAs laser, (c) Fatigue crack.

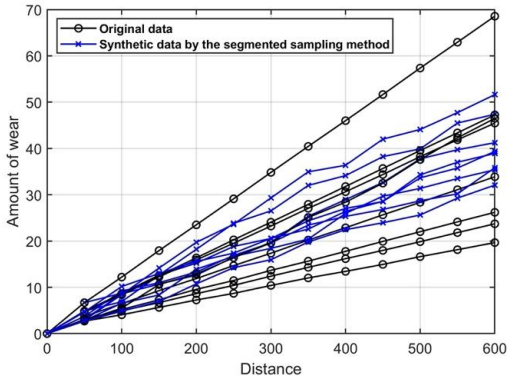
To begin with, the samples (#1-#8 for the first data set, #1-#12 for the second data set and #1-#16 for the third data set) are used to train the GMs. That is, the sample sizes for training are $n_1 = 8, n_2 = 12$ and $n_3 = 16$ respectively for these three data sets. Note that these data are trained and evaluated following the same methodology used in Section 5. For illustration, Figure 13 displays the original training sets and some generated synthetic data of the four GMs for the three degradation data sets. It can be seen that the degradation trend seems to be fully learned, and the synthetic data coincide broadly with the original data, indicating a satisfactory performance. To further evaluate the generative performance of the four GMs, the maximum mean discrepancy (MMD) is adopted as a criterion, which can measure the difference in distributions corresponding to the original data and the synthetic data. According to Gretton et al. [45], an empirical estimator of MMD can be written as



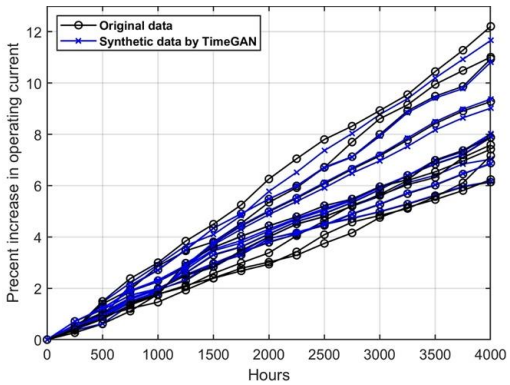
(a) (b)


Original data
Synthetic data by the diffusion model
Amount of wear
Distance

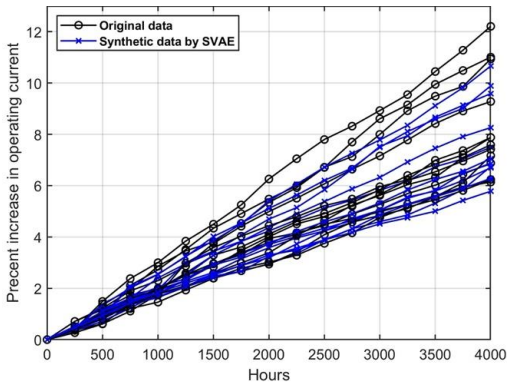
(c)


Original data
Synthetic data by the segmented sampling method
Amount of wear
Distance

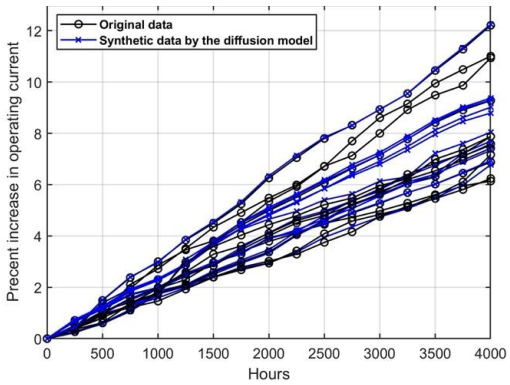
(d)


Original data
Synthetic data by TimeGAN
Precent increase in operating current
Hours

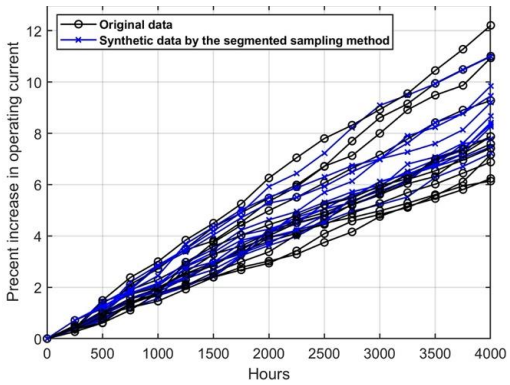
(e)


Original data
Synthetic data by SVAE
Precent increase in operating current
Hours

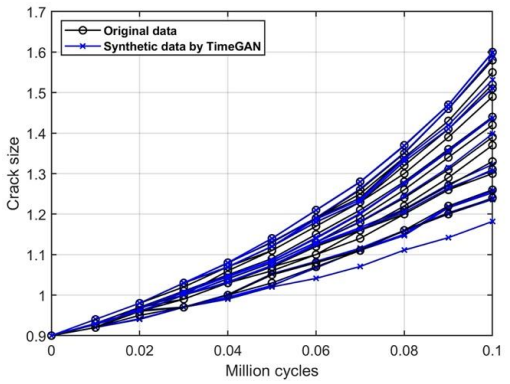
(f)


Original data
Synthetic data by the diffusion model
Precent increase in operating current
Hours

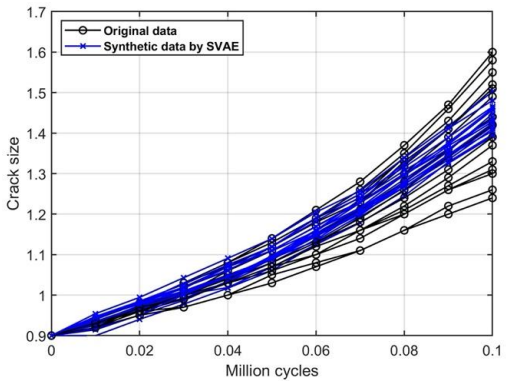
(g)


Original data
Synthetic data by the segmented sampling method
Precent increase in operating current
Hours

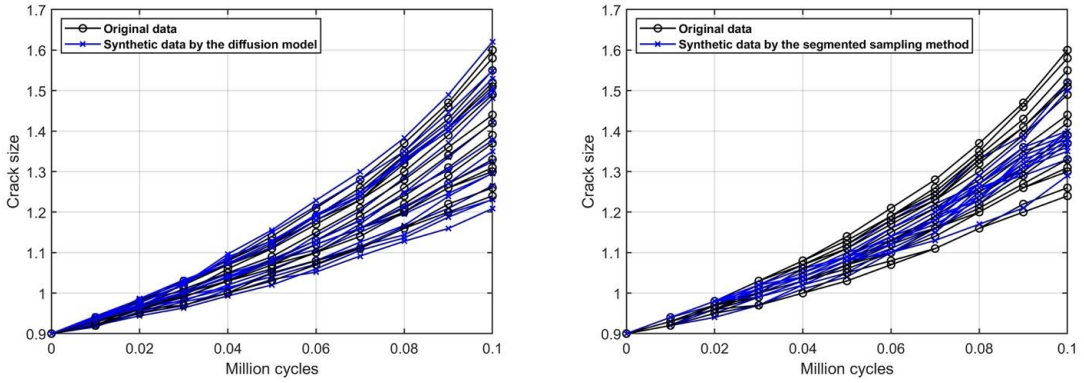
(h)


Original data
Synthetic data by TimeGAN
Crack size
Million cycles

(i)


Original data
Synthetic data by SVAE
Crack size
Million cycles

(j)



(k) (l)

Figure 13. Illustration of the original data and the synthetic data. (a)-(d) Train wheel, (e)-(h) GaAs laser, (i)-(l) Fatigue crack.

$$\mathrm{MMD}_{est,l} = \left[\frac{1}{n_l^2}\sum_{i_1,i_2=1}^{n_l} K(\mathcal{X}_{i_1},\mathcal{X}_{i_2}) - \frac{2}{n_l^2}\sum_{i_1,i_2=1}^{n_l} K(\mathcal{X}_{i_1},\widetilde{\mathcal{X}}_{i_2}) + \frac{1}{n_l^2}\sum_{i_1,i_2=1}^{n_l} K(\widetilde{\mathcal{X}}_{i_1},\widetilde{\mathcal{X}}_{i_2})\right]^{\frac{1}{2}},\ l = 1,2,3, \tag{30}$$

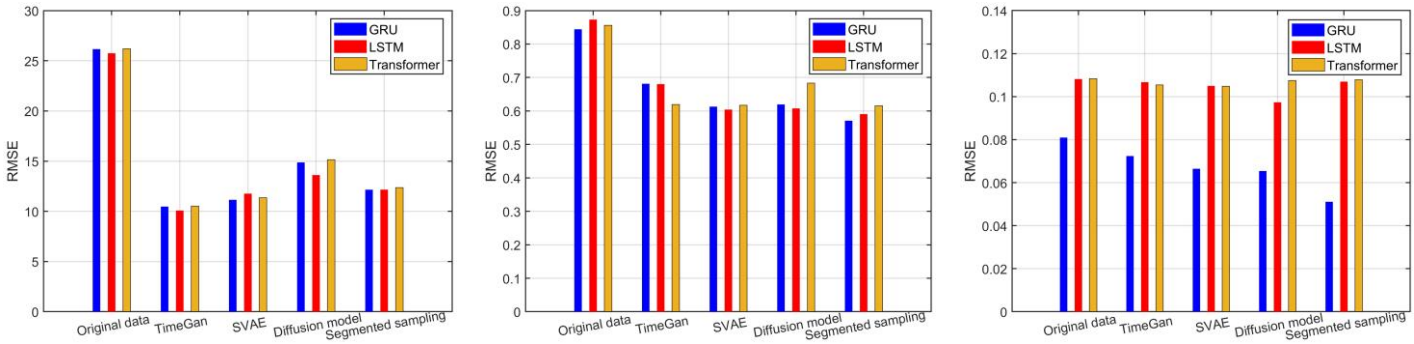
where $K(\cdot)$ is a kernel function, and $\mathcal{X}_i = (x_{i1}, x_{i2}, \cdots, x_{im})$ and $\widetilde{\mathcal{X}}_1$ are the original sample and the synthetic sample respectively. Further, we generate one hundred synthetic data sets with the same sample size (n_1, n_2 or n_3), and calculate an empirical estimator of MMD for each synthetic data set. The average values of one hundred MMD estimators corresponding to the four GMs are presented in Table 3.

Table 3. Comparison of the generative performance.

Data set	Average MMD value			
	TimeGAN	SVAE	Diffusion model	Segmented sampling
Train wheel	0.3757	0.3951	0.3617	0.5345
GaAs laser	0.2529	0.2472	0.2685	0.2461
Fatigue crack	0.2867	0.3652	0.3384	0.4033

It can be seen that they are comparable in the degradation generation of the case data: the diffusion model performs best for the train wheel degradation data set, while the segmented sampling method has a superior performance for the GaAs laser degradation data set, and TimeGAN generates the most similar data for the fatigue crack degradation data set.

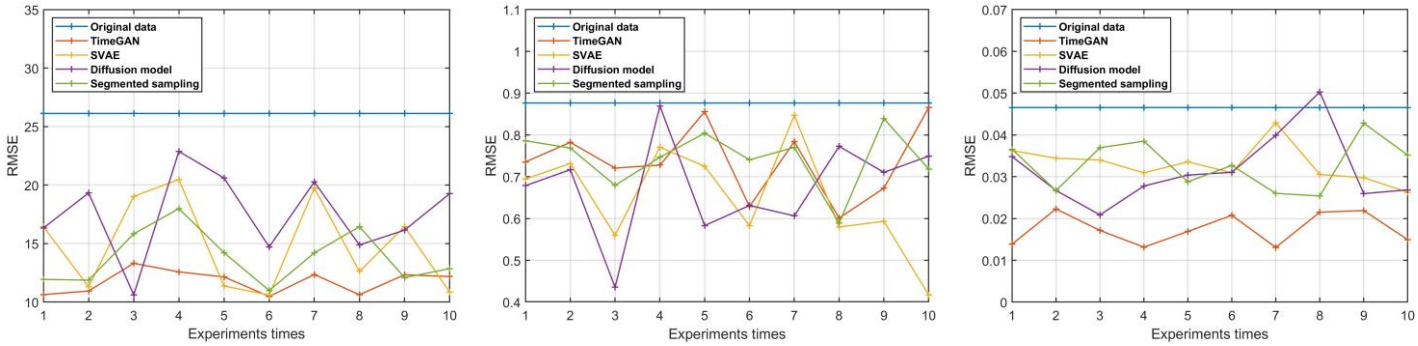
Then, the degradation values of units (#9-#11 for the first data set, #12-#15 for the second data set and #17-#20 for the third data set) at the last observation time points are predicted by the LSTM, GRU and Transformer networks, which are trained by utilizing the original training set and the synthetic data simultaneously. The resulting RMSEs are shown in Figure 14, where the prediction results without using the synthetic data are also presented. Clearly, when the synthetic data generated by any one of the four GMs are used, the resulting RMSEs are smaller than those based solely on the original training set, which also demonstrates that the four GMs have the capability of learning the essence of original degradation data. Additionally, despite the difference between RMSEs of the LSTM, GRU and Transformer networks in train wheel and laser data sets is small, that in crack data is noticeable, that is, the GRU network has better predictive performances than the LSTM and Transformer networks.



(a) (b) (c)

Figure 14. RMSEs of the LSTM, GRU and Transformer network. (a) Train wheel, (b) GaAs laser, (c) Fatigue crack.

Furthermore, since the original training data and the same amount of synthetic data are combined to train the predictive networks, it is imperative to delve into the stability of the degradation prediction when inputting different synthetic data. Consequently, experiments with different generation data are carried out on the three degradation data sets to explore the stability of the network. The prediction RMSEs of the GRU network are shown in Figure 15, presenting that the prediction errors of different degradation data are overall stable and smaller than those based solely on the original data. Therefore, it is verified that four GMs with GRU network are stable in the process of degradation generation and prediction.



(a) (b) (c)

Figure 15. The error results of the degradation prediction of 10 experiments with different degradation data. (a) Train wheel, (b) GaAs laser, (c) Fatigue crack.

7. Conclusion and Discussion

This paper aims at comparing several methods in terms of the degradation generation and prediction. Specifically, the TimeGAN, SVAE, diffusion model and a segmented sampling method are considered as four GMs to synthesize degradation data. Then, the LSTM, GRU, and Transformer networks are trained based on the original data and the extra synthetic data to predict degradation values. Both simulation studies and real data analyses are carried out for comparison, where the generative performance is evaluated by measuring the difference in distributions of the original data and the synthetic data, and the predictive performance is assessed by calculating the RMSEs between the actual degradation observations and their prediction values. From Table 1 and Table 3 and Figure 9 and Figure 14, we can see that the TimeGAN and the segmented sampling methods are in the top two many times. Further, the TimeGAN appears to be more robust than others, and the segmented sampling method has advantages in saving time because it does not need training. Thus, these two methods are recommended for degradation generation. Interestingly, the diffusion model, a newly considered method in the field of degradation generation, preforms competitive with them. In the meantime, it can be found that the GRU network has commensurate or smaller RMSEs in most cases compared with the LSTM and Transformer networks. Besides, the GRU network has a simplified architecture, resulting in lower computational costs than the LSTM and Transformer networks. As a result, the GRU network is recommended for degradation prediction. What is more, we can conclude that expansion of the original data by introducing the synthetic data is feasible and effective because the resulting predictive errors are reduced.

Meantime, generating scarce, multi-dimensional, and complex degradation data presents significant value but also poses challenges. In terms of multi-dimensional degradation data, we can categorize scenarios into two types: where different dimensions are independent and where they are correlated. For the former, the methods described in our paper can be straightforwardly applied to generate one-dimensional data for each dimension independently. However, the latter scenario presents greater challenges. Presented in Figure 10, Figure 11 and Table 2, SVAE, diffusion model and a segmented sampling method can be applied to the two-dimensional degradation generation and prediction. However, it is important to note that while implementing the TimeGAN model for generating degradation data, gamma noise needs to be added. This process is straightforward in single-dimensional scenarios but poses potential challenges in multi-dimensional settings, requiring more in-depth theoretical analysis to ascertain feasibility and identifying an area for further research.

In the future, further applications of synthetic data in RUL prediction, maintenance optimization, burn-in testing and others

will be explored. In addition, how to integrate some prior information or physical mechanism about the degradation data into the existing GMs to enhance the quality of synthetic data is worth investigating.

Acknowledgments

We would like to thank the reviewers for their insightful comments and suggestions, which have significantly improved the quality of this paper.

References

1. Ye Z, Xie M. Stochastic modelling and analysis of degradation for highly reliable products. Appl Stoch Models Bus & Ind 2015;31:16–32. https://doi.org/10.1002/asmb.2063.
2. Meeker WQ, Escobar LA, Pascual FG. Statistical methods for reliability data. Second edition. Hoboken, NJ: Wiley; 2022.
3. Weerahandi S. Generalized Confidence Intervals. Exact Statistical Methods for Data Analysis, New York, NY: Springer New York; 1995, p. 143–68. https://doi.org/10.1007/978-1-4612-0825-9_6.
4. Chen P, Ye Z, Xiao X. Pairwise model discrimination with applications in lifetime distributions and degradation processes. Naval Research Logistics 2019;66:675–86. https://doi.org/10.1002/nav.21875.
5. Wang X, Wang BX, Wu W, Hong Y. Reliability analysis for accelerated degradation data based on the Wiener process with random effects. Quality & Reliability Eng 2020;36:1969–81. https://doi.org/10.1002/qre.2668.
6. Song K, Cui L. Fiducial inference-based failure mechanism consistency analysis for accelerated life and degradation tests. Applied Mathematical Modelling 2022;105:340–54. https://doi.org/10.1016/j.apm.2021.12.048.
7. Chen P, Ye Z-S. Uncertainty quantification for monotone stochastic degradation models. Journal of Quality Technology 2018;50:207–19. https://doi.org/10.1080/00224065.2018.1436839.
8. Lu Y, Shen M, Wang H, Wang X, van Rechem C, Fu T, et al. Machine Learning for Synthetic Data Generation: A Review 2024.
9. De S, Bermudez-Edo M, Xu H, Cai Z. Deep Generative Models in the Industrial Internet of Things: A Survey. IEEE Trans Ind Inf 2022;18:5728–37. https://doi.org/10.1109/TII.2022.3155656.
10. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, Cambridge, MA, USA: MIT Press; 2014, p. 2672–80.
11. Kingma DP, Welling M. An Introduction to Variational Autoencoders. Found Trends Mach Learn 2019;12:307–92. https://doi.org/10.1561/2200000056.
12. Liu J, Qu F, Hong X, Zhang H. A Small-Sample Wind Turbine Fault Detection Method With Synthetic Fault Data Using Generative Adversarial Nets. IEEE Transactions on Industrial Informatics 2019;15:3877–88. https://doi.org/10.1109/TII.2018.2885365.
13. Liu C, Antypenko R, Sushko I, Zakharchenko O. Intrusion Detection System After Data Augmentation Schemes Based on the VAE and CVAE. IEEE Transactions on Reliability 2022;71:1000–10. https://doi.org/10.1109/TR.2022.3164877.
14. Booyse W, Wilke DN, Heyns S. Deep digital twins for detection, diagnostics and prognostics. Mechanical Systems and Signal Processing 2020;140:106612. https://doi.org/10.1016/j.ymssp.2019.106612.
15. Yu H, Wang QF, Shi JY. Data Augmentation Generated by Generative Adversarial Network for Small Sample Datasets Clustering. Neural Process Lett 2023;55:8365–84. https://doi.org/10.1007/s11063-023-11315-z.
16. Zhang W, Li Z, Li G, Zhuang P, Hou G, Zhang Q, et al. GACNet: Generate Adversarial-Driven Cross-Aware Network for Hyperspectral Wheat Variety Identification. IEEE Trans Geosci Remote Sensing 2024;62:1–14. https://doi.org/10.1109/TGRS.2023.3347745.
17. Zhang W, Zhao W, Li J, Zhuang P, Sun H, Xu Y, et al. CVANet: Cascaded visual attention network for single image super-resolution. Neural Netw 2024;170:622–34. https://doi.org/10.1016/j.neunet.2023.11.049.
18. Shangguan A, Xie G, Fei R, Mu L, Hei X. Train wheel degradation generation and prediction based on the time series generation adversarial network. Reliability Engineering & System Safety 2023;229:108816. https://doi.org/10.1016/j.ress.2022.108816.
19. Yoon J, Jarrett D, van der Schaar M. Time-series generative adversarial networks. Proceedings of the 33rd International Conference on Neural Information Processing Systems, Red Hook, NY, USA: Curran Associates Inc.; 2019.
20. Shangguan A, Feng N, Mu L, Fei R, Hei X, Xie G. SVAE-GRU-based degradation generation and prediction for small samples. Quality & Reliability Eng 2023;39:2851–68. https://doi.org/10.1002/qre.3394.

21. Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. Proceedings of the 34th International Conference on Neural Information Processing Systems, Red Hook, NY, USA: Curran Associates Inc.; 2020.
22. Ke W, Guo Y, Liu Q, Chen W, Wang P, Luo H, et al. MDM: Meta diffusion model for hard-constrained text generation. Knowledge-Based Systems 2024;283:111147. https://doi.org/10.1016/j.knosys.2023.111147.
23. Zeng J, Liu X, Li Z. Radio Anomaly Detection Based on Improved Denoising Diffusion Probabilistic Models. IEEE Communications Letters 2023;27:1979–83. https://doi.org/10.1109/LCOMM.2023.3290813.
24. Dhariwal P, Nichol A. Diffusion Models Beat GANs on Image Synthesis. In: Ranzato M, Beygelzimer A, Dauphin Y, Liang PS, Vaughan JW, editors. Advances in Neural Information Processing Systems, vol. 34, Curran Associates, Inc.; 2021, p. 8780–94.
25. Gupta P, Hayat M, Dhall A, Do T-T. Conditional Distribution Modelling for Few-Shot Image Synthesis with Diffusion Models. ArXiv 2024;abs/2404.16556.
26. Wang Z, Zhang H. Customized Load Profiles Synthesis for Electricity Customers Based on Conditional Diffusion Models. IEEE Trans Smart Grid 2024:1–1. https://doi.org/10.1109/TSG.2024.3366212.
27. Liu Y, Shen R, Shen X. Novel Uncertainty Quantification through Perturbation-Assisted Sample Synthesis. IEEE Transactions on Pattern Analysis and Machine Intelligence 2024:1–12. https://doi.org/10.1109/TPAMI.2024.3393364.
28. Shen X, Liu Y, Shen R. Boosting Data Analytics with Synthetic Volume Expansion n.d.
29. Efron B, Tibshirani RJ. An Introduction to the Bootstrap. 0 ed. Chapman and Hall/CRC; 1994. https://doi.org/10.1201/9780429246593.
30. Zhou S, Xu A. Exponential Dispersion Process for Degradation Analysis. IEEE Transactions on Reliability 2019;68:398–409. https://doi.org/10.1109/TR.2019.2895352.
31. Guo J, Wang C, Cabrera J, Elsayed EA. Improved inverse Gaussian process and bootstrap: Degradation and reliability metrics. Reliability Engineering & System Safety 2018;178:269–77. https://doi.org/10.1016/j.ress.2018.06.013.
32. Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Computation 1997;9:1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.
33. Cho K, Merrienboer B van, Gülçehre Ç, Bahdanau D, Bougares F, Schwenk H, et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. Conference on Empirical Methods in Natural Language Processing, 2014. https://doi.org/10.3115/v1/D14-1179.
34. Yang K, Wang Y, Yao Y, Fan S. Remaining useful life prediction via long-short time memory neural network with novel partial least squares and genetic algorithm. Quality and Reliability Engineering International 2021;37:1080–98. https://doi.org/10.1002/qre.2782.
35. Ungurean L, Micea MV, Cârstoiu G. Online state of health prediction method for lithium-ion batteries, based on gated recurrent unit neural networks. International Journal of Energy Research 2020;44:6767–77. https://doi.org/10.1002/er.5413.
36. Lin M, You Y, Wang W, Wu J. Battery health prognosis with gated recurrent unit neural networks and hidden Markov model considering uncertainty quantification. Reliability Engineering & System Safety 2023;230:108978. https://doi.org/10.1016/j.ress.2022.108978.
37. Sun R, Yang Z, Yang L, Qiao B, Chen X, Gryllias KC. Planetary gearbox spectral modeling based on the hybrid method of dynamics and LSTM. Mechanical Systems and Signal Processing 2020;138:106611. https://doi.org/10.1016/j.ymssp.2019.106611.
38. Gu L, Zheng R, Zhou Y, Zhang Z, Zhao K. Remaining useful life prediction using composite health index and hybrid LSTM-SVR model. Quality and Reliability Engineering International 2022;38:3559–78. https://doi.org/10.1002/qre.3151.
39. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA: Curran Associates Inc.; 2017, p. 6000–10.
40. Wen Q, Zhou T, Zhang C, Chen W, Ma Z, Yan J, et al. Transformers in Time Series: A Survey. International Joint Conference on Artificial Intelligence, 2022. https://doi.org/10.24963/ijcai.2023/759
41. Lee S, Hong J, Liu L, Choi W. TS-Fastformer: Fast Transformer for Time-series Forecasting. ACM Trans Intell Syst Technol 2024;15. https://doi.org/10.1145/3630637.
42. Song K, Cui L. A common random effect induced bivariate gamma degradation process with application to remaining useful life prediction. Reliability Engineering & System Safety 2022;219:108200. https://doi.org/10.1016/j.ress.2021.108200.
43. Freitas MA, de Toledo MLG, Colosimo EA, Pires MC. Using degradation data to assess reliability: a case study on train wheel degradation. Quality and Reliability Engineering International 2009;25:607–29. https://doi.org/10.1002/qre.995.

44. Peng C-Y, Cheng Y-S. Student-t Processes for Degradation Analysis. Technometrics 2020;62:223–35. https://doi.org/10.1080/00401706.2019.1630008.
45. Gretton A, Borgwardt KM, Rasch MJ, Scholkopf B, Smola A. A Kernel Two-Sample Test. Journal of Machine Learning Research 2012;13:723–73.

Appendix

Table 4 provides the hyper-parameter configurations of the three deep learning-based GMs and two predictive networks.

Table 4 Summary of the hyper-parameters of the GMs and predictive networks.

Network	Hyper-parameters	Values							Activation Function
		Numerical simulation				Case study			
		(10,10)	(10,20)	(20,10)	(20,20)	Train wheel	GaAs laser	Fatigue crack	
TimeGAN	Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	-
	iteration	3000	3000	3000	3000	5000	5000	5000	-
	Batch size	32	32	32	32	16	16	16	-
	Num of layers	3	3	3	3	3	3	3	-
	Hidden dimension	24	24	24	24	24	24	24	-
	Network of generators	GRU	GRU	GRU	GRU	GRU	GRU	GRU	Sigmoid
	Network of discriminators	GRU	GRU	GRU	GRU	GRU	GRU	GRU	Sigmoid
	Noise distribution	Gamma(2, 1)	Gamma(2, 1)	Gamma(2, 1)	Gamma(2, 1)	Gamma(0.5, 22)	Gamma(0.5, 22)	Gamma(0.5, 22)	-
SVAE-GRU	Number of the VAE	2	2	2	2	2	2	2	-
	Epoch	200	200	200	200	200	200	300	-
	Learning rate	5e-4	5e-4	5e-4	5e-4	5e-4	5e-4	5e-4	-
	Batch size	16	16	16	16	16	16	16	-
	GRU units	50	50	50	50	50	50	50	-
	Fully connected layer	2	2	2	2	2	2	2	ReLU
	Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	-
	Loss function	MAELoss	MAELoss	MAELoss	MAELoss	MAELoss	MAELoss	MAELoss	-
Diffusion model	Linear layer	3	3	3	3	3	3	3	ReLU
	Embedding layer	2	2	2	2	2	2	2	-
	Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	-
	Iteration	30000	70000	50000	190000	140000	80000	86000	-
	Batch size	32	32	32	32	32	32	32	-
	Num of steps	200	500	200	500	1800	800	100	-
LSTM/GRU	Units	3	3	3	3	3	3	3	-
	Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	-
	Loss function	MSELoss	MSELoss	MSELoss	MSELoss	MSELoss	MSELoss	MSELoss	-
	Output	1	1	1	1	1	1	1	Sigmoid
Transformer	Units	1	1	1	1	1	1	1	-
	Loss function	MSELoss	MSELoss	MSELoss	MSELoss	MSELoss	MSELoss	MSELoss	-
	Output	1	1	1	1	1	1	1	-

The code of the main part of the experiment will be made available https://github.com/Lixt2000/Degradation-generation-and-prediction.