Synthetic Data Augmentation for Aircraft Maintenance Prognostics using a Supervised Spatial Temporal Generative Adversarial Network

Daniël Timmermans
[a]*Faculty of Aerospace Engineering, Delft University of Technology, HS 2926 Delft, The Netherlands*

Abstract

The performance of Remaining Useful Life (RUL) prediction models is often limited by data scarcity, especially in safety-critical systems like aircraft engines where failure data is rare. To address this challenge, we propose the Super-SpaceTime GAN, a framework for generating synthetic condition monitoring (CM) data to enhance RUL predictions. The framework incorporates dual-conditioning on operating conditions (OCs) and RUL labels, an autoencoder-based latent space for denoising, and a supervised loss function to align synthetic data with real degradation trajectories. Evaluated on the CMAPSS FD004 dataset, the Super-SpaceTime GAN generates synthetic data that closely mimic real distributions, as verified using Jensen-Shannon Distance, Principal Component Analysis, t-distributed Stochastic Neighbor Embedding, and a novel autoencoder-based health monitoring metric. The framework demonstrates improvement in prognostic performance in limited training data scenarios, with gains persisting even in data-rich settings. These findings highlight the potential of the Super-SpaceTime GAN to improve RUL predictions by addressing data scarcity, making it a valuable tool for Prognostics and Health Management (PHM).

1 Introduction

In the aerospace industry and, more specifically, for engine maintenance, Prognostics and Health Management (PHM) plays a pivotal role in ensuring the safety and reliability of critical systems. However, a significant challenge in PHM for engine maintenance is the limited availability of failure data, since operating engines until failure is impractical and highly unsafe. This data scarcity hinders the development of remaining useful life (RUL) models, which are essential for predicting when engine maintenance should be performed. To tackle data scarcity, Generative Adversarial Networks (GANs) offer a promising solution by generating synthetic data that mimic real-world failure scenarios. Synthetic data generated by GANs can augment existing training datasets, providing additional, but realistic, failure scenarios that improve the accuracy and robustness of RUL prognostic models.

A particularly promising scenario for GAN-based data augmentation arises when multiple airliners, each with a limited availability of failure datasets, collectively train a single GAN model. For instance, suppose five airliners each have condition monitoring data for 10 engines, resulting in a combined pool of 50 engines. By training a single GAN on this combined training set, each airliner could employ the GAN for synthetic data augmentation on its own smaller dataset, increasing RUL model predictive performance for its RUL models. Importantly, this approach enables the GAN to be trained on a sufficiently large dataset while airliners do not have to share their complete datasets with others. Solely the entity that trains the GAN needs full access to all the datasets, such as the engine manufacturer. This setup illustrates how GANs can be instrumental in collaboratively overcoming data scarcity without issues of sharing data, which is a fundamental improvement in a safety-critical industry like aerospace.

First introduced by Goodfellow et al. (2014) [1], GANs represented a breakthrough in synthetic data generation. In GANs, the discriminator and generator, both deep neural networks (DNNs), work together to generate data that follows the distribution of the original data as much as possible. The ingenuity of GANs is that there is a corrective feedback loop from the discriminator to the generator, which consists of the discriminator's prediction of whether the data it received are fake or real. This corrective feedback helps the generator recognize which generated data samples were successful or not in fooling the discriminator, aiding the generator in training. GANs have been widely adopted in domains such as image synthesis [2], text generation [3], and video modeling [4]. While GANs originally emerged in the computer vision domain, their use in generating time series data has been growing in recent years [5]. More recently, GANs have been adapted for generating time-series data, with frameworks such as C-RNN-GAN proposed by Mogren [6], and TimeGAN developed by Yoon et al. [7], emerging as powerful tools for modeling complex temporal dependencies [8]–[11].

Despite their potential, timeseries GANs, called temporal GANs, are notoriously difficult to train, often suffering from mode collapse and disappearing gradients [12]–[14]. Additionally, these methods frequently fall short when applied to complex engineered systems like aircraft engines, where complicated physical processes such as health degradation under varying operating conditions (OCs) must be modeled. OCs refer to the diverse environmental and operational factors affecting engine performance, such as altitude, flight velocity, and engine load. These conditions can vary significantly across the different phases of flight (e.g. takeoff, cruise, landing, and taxiing), and therefore have a large influence on the overall health trajectory of the engine. Accurately capturing the effects of OCs is essential to ensure the synthetic data aligns with the health degradation patterns observed in real-world data. While significant progress has been made in utilizing temporal GANs in the PHM domain [15], many challenges remain unsolved. This leads to the central research question explored in this work: **Can a GAN model generate augmented datasets that improve the predictive performance of a neural network in estimating the remaining useful life?**

Recent approaches have tailored GANs to adapt to PHM applications. Lang et al. [16] demonstrated the feasibility of vanilla GANs for augmenting RUL models, but limited their study to a dataset without varying OCs and failure modes. Zhang et al. [17] developed a convolutional recurrent GAN for data augmentation for RUL prognostics. He et al. [18] proposed a semi-supervised

GAN regression model for jet engine RUL prediction, incorporating suspension history for improved prognostics. Rombach et al. [19] introduced a Wasserstein GAN designed for the controlled generation of previously unseen faults, enabling a transfer of fault signatures between two different failure domains if they shared a healthy data class. Zhou et al. [20] proposed a framework leveraging a GAN to generate extra fault features, with a discriminator tasked to filter out unqualified fault samples. Yan et al. [21] proposed a variational autoencoder-based conditional Wasserstein GAN with gradient penalty (CWGAN-GP-VAE) to diagnose various faults for chillers. Xiong et al. [22] extended the TimeGAN framework by incorporating a controlled physics-informed GAN. This approach employed a physics-informed loss function based on system health indicators inferred by a surrogate model, ensuring that the generated synthetic data closely mirrored the health degradation trajectory in the real data.

Building upon these advancements, we propose a novel temporal GAN-based framework, the Super-SpaceTime GAN, designed specifically for generating synthetic condition monitoring (CM) data to enhance RUL predictions. Traditional temporal GANs often struggle with capturing complex temporal and spatial dependencies, especially in scenarios involving health degradation under multiple operational conditions. The Super-SpaceTime GAN tackles these challenges through:

1) **An autoencoder-based latent space representation** to denoise the data and allow the generator and discriminator to focus on critical features, streamlining the notoriously challenging GAN training process.
2) **A Supervised loss function** to guide the generator in producing realistic outputs that align with both temporal and spatial dependencies.
3) **Dual conditioning on RUL labels and OCs** to ensure that the synthetic data accurately reflect real-world health degradation dynamics.

We evaluate the Super-SpaceTime GAN using the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) FD004 dataset, a widely adopted benchmark for RUL prediction tasks. This dataset represents a challenging scenario with multiple failure modes and OCs. Our evaluation comprises of two comprehensive case studies:

- **Synthetic data quality assessment:** Using metrics such as the Jenson-Shannon Distance, Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE), and a novel autoencoder-based health degradation monitoring approach, we assess the quality of the generated synthetic data.
- **Prognostic Performance Evaluation:** We examine the impact of synthetic data augmentation on the performance of a state-of-the-art RUL model. We demonstrate improved prognostic accuracy of RUL models in data-scarce scenarios where the GAN, also trained on a limited dataset, generates synthetic data to enhance model performance. Additionally, we show that the proposed GAN facilitates data augmentation that boosts prognostic performance even in scenarios where ample real data is available.

This research highlights the potential of using GANs for augmenting datasets in PHM. Our findings provide strong evidence that synthetic data could be used to improve RUL predictions. Furthermore, the challenges encountered during data quality assessment provide valuable insights for future research, laying the foundations for further improvements in the utility of synthetic data in PHM. The remainder of this paper is organized as follows. Section 2 presents the details of the proposed Super-SpaceTime GAN framework. In Section 3, the CMAPSS dataset used for the case studies is introduced. Section 4 outlines the data quality assessment methodology and presents its findings, while Section 5 details the prognostic performance evaluation of synthetic data, including the methodologies used and the corresponding results. Finally, conclusions are provided in Section 6, leading to the directions for future research presented in Section 7. For replication, the complete code for methodology and experiments conducted in this paper are available at https://github.com/DanielTimmermans.

2 Methodology

This section outlines the methodology employed to address the research question. It is structured as follows: Section 2.1 provides an overview of the vanilla GAN framework, highlighting its core principles. Next, Section 2.2 examines the TimeGAN architecture, which serves as a foundational component for the proposed GAN. In Section 2.3, modifications to the TimeGAN are introduced, specifically designed to enhance its capability for data augmentation in PHM applications. Finally, Section 2.4 introduces the Jenson-Shannon Distance as a key metric for assessing the quality of synthetic data during training.

2.1 Generative Adversarial Network

The Generative Adversarial Network (GAN) consists of two main components: a generator (G) and a discriminator (D). The generator's goal is to produce synthetic data samples ($\hat{x}$) that follow the distribution of real data samples (x). Meanwhile, the discriminator acts as a classifier, outputting $P(x)$, the probability that a given real data sample x is correctly identified as real, and $P(\hat{x})$, the probability that a synthetic data sample $\hat{x}$ is incorrectly identified as real. The GAN's training process involves a two-player Min-Max game where the generator strives to fool the discriminator with its generated data, and the discriminator attempts to identify real versus synthetic data correctly. Both the generator and discriminator have trainable parameters, Θ_G and Θ_D respectively, which are updated during training to improve their performance. The training dynamics of this adversarial process are depicted in Figure 1.

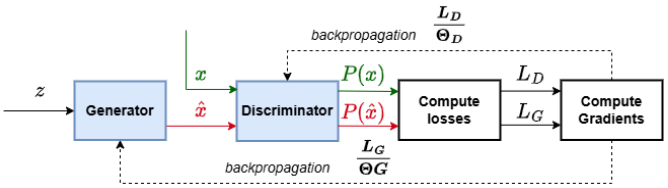


Fig. 1: The training dynamics of a Vanilla GAN.

Training commences by sampling random noise vectors z from a predefined random distribution, ensuring the generator produces a diverse output. These random noise vectors z are fed to the generator, which uses the random noise vectors to generate a synthetic data sample $\hat{x}$ aiming to mimic the distribution of a real data sample x as closely as possible. Following, the synthetic data samples $\hat{x}$ and real data samples x are fed to the discriminator simultaneously. The discriminator evaluates each sample individually and computes the probabilities $P(x)$ and $P(\hat{x})$ that the real data x and synthetic data $\hat{x}$ are deemed real (1 meaning that the discriminator is convinced that the data sample is real and probability close to 0 meaning that the discriminator is confident that the data sample is synthetic). From these probabilities, the discriminator loss L_D is calculated based on how well the discriminator can distinguish between real and synthetic data. The Generator loss L_G on the other hand, measures how well the generator succeeded in fooling the discriminator into thinking that its fake data are deemed real by the discriminator. Finally, in the backpropagation, the discriminator's parameters Θ_D and the generator's parameters Θ_G are updated by minimizing the losses L_G and L_D respectively. This encourages the generator to create more realistic data to deceive the discriminator and enhances the discriminator's ability to distinguish between real and synthetic data. This adversarial training process proceeds iteratively, with the generator continuously improving the quality of its synthetic data and

the discriminator enhancing its ability to distinguish between real and synthetic data. The GAN's objective is mathematically depicted:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))], \quad (1)$$

which the discriminator aims to maximize. It tries to maximize $D(x)$, i.e. the probability $P(x)$ that it deems the real data real; and on the other hand, minimize $D(G(z))$ i.e. the probability $P(\hat{x})$ that it classifies the synthetic data as real. The generator has a competing objective, as it aims to minimize Eq. 1. It does this by maximizing $D(G(z))$ and thus the probability $P(\hat{x})$ that the discriminator deems the synthetic data $\hat{x}$ real, which is exactly opposite to the discriminator's objective. Since the generator has no explicit influence on the classification of the discriminator on the real data samples, the generator only actively minimizes the second part of the equation. Over time if the generator succeeds in generating more realistic synthetic data, the Min-Max game reaches an equilibrium point where the discriminator's output is approximately equal to 0.5, meaning that it is unable to distinguish between real and synthetic data.

2.2 TimeGAN Framework

The TimeGAN, developed by Yoon et al. (2019) [7], provides the underlying framework for our proposed Super-SpaceTime GAN. The TimeGAN integrates the strengths of GANs, convolutional neural networks (CNNs), and autoencoders (AEs) to generate realistic time series data. TimeGAN's primary novelties are twofold: a supervised loss enhances the modeling of temporal dynamics and an embedding network that provides a more focused adversarial learning space.

2.2.1 Embedding Network

The embedding network plays a crucial role within the SpaceGAN framework by learning accurate and efficient data representations of the original time-series data. The embedding network consists of an autoencoder. The autoencoder consists of an encoder that compresses the high-dimensional time-series data into a latent space that captures the essential features of the original data, and a decoder that reconstructs the latent data back to the original time-series data. Encoding the original high-dimensional feature space into a more focused latent space representation allows the GAN to focus on the most important features by reducing noise and irrelevant features. This significantly simplifies the GAN's training process as it works in a more focused latent space, making it easier for the generator to generate synthetic data that follows the underlying distribution of the original dataset. Furthermore, the discriminator, which operates in the same latent space, distinguishes between fake and synthetic data more easily as it solely focuses on key data features and has to deal with less noise.

2.2.2 Embedding Space & Latent Space

TimeGAN processes data through two distinct representation spaces. The embedding space performs the initial transformation of raw input data into basic feature representations. This space focuses on capturing fundamental patterns while reducing noise, essentially creating a "rough sketch" of the data's structure. Embedding the data transforms data from its raw dimensionality to a higher-dimensional space where patterns are more easily distinguishable. The latent space then further processes these embeddings through multiple convolutional layers to capture temporal dynamics in greater depth. While the dimensionality stays identical, the latent representations contain richer temporal relationships due to deeper processing. Both the autoencoder and generator follow this procedure: the autoencoder transforms real data into an embedding space, which is subsequently mapped to the latent space for reconstruction. Similarly, the generator creates synthetic data by first creating embeddings and generates them them into latent space. This parallel structure ensures that both real and synthetic data undergo similar processing; enabling direct comparison of real embeddings with synthetic embeddings, and real latent space to synthetic latent space.

Figure 3 visualizes how the TimeGAN framework processes samples through the two representation spaces. First, raw input data x_t enters the embedding space via the *Encoder Embedder*, yielding the real embeddings e_{x_t} (highlighted in red). In parallel, synthetic inputs z_t, sampled from a noise distribution, pass through the *Generator Embedder* which produces synthetic embeddings e_{z_t} (also shown in red). Next, the model further processes these embeddings into the latent space by passing them through CNN layers (the *Encoder* for real data and *Generator* for synthetic data), yielding the latent vectors h_{x_t} (real) and h_{z_t} (synthetic), both highlighted in green.

Because both the autoencoder and the generator follow this two-step embedding-to-latent procedure, the discriminator can directly compare:

- **Embeddings:** real e_{x_t} vs. synthetic e_{z_t}
- **Latent vectors:** real h_{x_t} vs. synthetic h_{z_t}

This parallel structure allows the discriminator to learn the real data distributions through embeddings and latent vectors in the same representational spaces used by the generator's synthetic data. Because the real and synthetic data are classified in a more focused space, the discriminator can perform a more focused comparison. This ultimately improves the generator's output as the discriminator becomes better in its classification and forces the generator to produce more realistic synthetic samples. Furthermore, the generator operates in a more structured space, allowing it to focus more on generating high-quality synthetic samples.

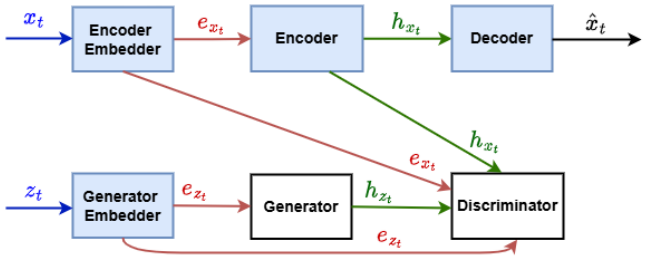


Fig. 2: Diagram highlighting the embedding- and latent spaces of the TimeGAN framework.

2.2.3 TimeGAN Training Dynamics

To effectively learn the distribution of the original time series, TimeGAN employs a combination of adversarial training, supervised loss, and reconstruction loss, ensuring that both the generator and embedding network capture meaningful temporal dependencies. The first key component of the TimeGAN is the **reconstruction loss**, which trains the autoencoder. This loss is defined as: Eq. 2:

$$L_R = \mathbb{E}_{x_{1:T}\sim p}\left[\sum_t \|x_t - \tilde{x}_t\|_2\right], \quad (2)$$

with x_t and $\tilde{x}_t$ sampled from the actual- and reconstructed data (i.e. the output of the decoder) respectively, which are in the original feature dimension corresponding to the sensor readings. The subscript t in x_t denotes the time dimension, indicating that the samples are sequential and reflect sensor measurements captured at a specific timestep. Each x_t has multiple dimensions, where each dimension represents a distinct sensor feature. The $x_{1:T\sim p}$ subscript indicates that the sequences $x_{1:T}$ are drawn from the true data distribution p. This objective function aims to guide the autoencoder's reconstructed data samples $\tilde{x}_t$ to mimic the real data distribution of x_t as closely as possible.

The second objective function is the **unsupervised (adversarial) loss**, which is the classic GAN loss (Eq. 1) that governs the Min-Max

Synthetic Data Augmentation for Aircraft Maintenance

Prognostics using a Supervised Spatial Temporal Generative Adversarial Network

by

Daniël Timmermans

To obtain the degree of Master of Aerospace Engineering
at the Delft University of Technology,
to be defended publicly 20-2-2025

I.I. de Pater	TU Delft
Kristupas Bajarunas	Zurich University of Applied Scienc
M.J. Ribeiro	TU Delft
N. Eleftheroglou	TU Delft

TU Delft

game used for training in GANs:

$$L_U = \mathbb{E}_{x_{1:T}\sim p}\left[\sum_t \log y_t\right] + \mathbb{E}_{x_{1:T}\sim \hat{p}}\left[\sum_t \log(1-\hat{y}_t)\right]. \quad (3)$$

Here, $x_{1:T\sim p}$ and $x_{1:T\sim \hat{p}}$ represent sequences $x_{1:T}$ drawn from the true data distribution p and the learned data distribution $\hat{p}$ respectively. The discriminator produces outputs y_t and $\hat{y}_t$, which classify the real data and synthetic data as real or fake at both the embedding and latent space levels. Since both real data and synthetic data are processed through these two spaces, the discriminator evaluates the synthetic data quality at both levels, producing y_t for real data and $\hat{y}_t$ for synthetic data in both the embedding and latent representations. In the embedding space, it classifies the synthetic embeddings (e_{z_t}) generated from random noise and the real embeddings (e_{x_t}). This comparison is crucial as the embeddings from random noise can not be evaluated using the reconstruction loss. Similarly, in the latent space, the discriminator compares the synthetic latent vectors (h_{z_t}) against real latent vectors (h_{x_t}). This multi-level discrimination ensures that the generator receives feedback on both its initial embedding and subsequent temporal processing in the latent space.

The adversarial feedback may not be sufficient for the generator to generate synthetic data that captures the stepwise conditional distributions of the data. Therefore, the final objective function is introduced as the **supervised loss**:

$$L_S = \mathbb{E}_{x_{1:T}\sim p}\left[\sum_t \|h_{x_t} - \hat{h}_{x_{t-1}}\|_2\right]. \quad (4)$$

Here, next to the adversarial training, TimeGAN trains in a closed-loop mode. The generator receives sequences of actual embeddings e_{x_t} and processes these embeddings to generate conditional synthetic latent vectors $\hat{h}_{x_t}$. Then, the conditional synthetic latent vectors at time step $t-1$ ($\hat{h}_{x_{t-1}}$) are compared to the actual latent vectors at t (h_{x_t}) using Mean Squared Error. The comparison of consecutive time steps trains the generator to predict the next timestep's latent representation, ensuring it learns the temporal progression of the data. The supervised loss thus ensures that the generator learns to capture the temporal dynamics of the real data.

The three aforementioned losses govern the backpropagation of the training epochs. For visual aid, the TimeGAN's structure is visually depicted in Figure 3. In the diagram, the six model parts are showcased in light blue. The input data is depicted with dark blue arrows, and the overall data flow between model parts is depicted with black arrows. The green arrow is the adversarial loss computed by the discriminator's real and fake data classifications. The data flow between model parts where reconstructed/synthetic data are compared with actual data, including latent vectors, are depicted in red and used to determine the reconstruction loss and supervised loss. Below, the training dynamics of each of the six TimeGAN model parts are elaborated.

- **Encoder Embedder:** The encoder embedder compresses the input data x into the embedded representation e_{x_t}, which is fed to the encoder and generator for further processing, and to the discriminator for classification such that it can learn the distribution of the embedded actual data.
- **Encoder:** Takes as input the actual embedding e_{x_t} and processes it further to generate the actual latent vectors h_{x_t}. These vectors are supplied to the decoder such that $\tilde{x}_t$ can be reconstructed. For the supervised loss L_S, h_{x_t} is compared with the conditional latent vectors $\hat{h}_{x_t}$ using the Mean Squared Error. Furthermore, the real latent vectors h_{x_t} are fed to the discriminator to allow it to learn the distribution of an actual latent vector.
- **Decoder:** The primary function of the decoder is to reconstruct the actual latent vectors h_{x_t} into the reconstructed data x. The reconstructed data $\hat{x}$ are then compared to the actual data x using the MSE as the reconstruction loss L_R. The decoder must be trained properly to accurately reconstruct the generator's synthetic data from the latent space back to the original feature space during the final data generation.
- **Generator Embedder:** Takes random noise samples z as input and processes them into the synthetic embeddings e_{z_t}, which are fed to the generator for further processing and to the discriminator for classification.
- **Generator:** Takes as input the synthetic embeddings e_{z_t} and generates the synthetic latent vectors h_{z_t} to feed to the discriminator for classification. This is the primary goal of the whole GAN as the synthetic data is generated here. Furthermore, the generator takes the actual embeddings e_{x_t} to generate the conditional synthetic latent vectors $\hat{h}_{x_t}$, which are compared with the actual latent vectors h_{x_t} to determine the supervised loss L_S. This second generator objective aims to train the generator to produce synthetic data in the latent space that aligns with the actual data distribution.
- **Discriminator:** The discriminator's role is to differentiate between actual and synthetic data in both the embedding and latent spaces. It takes as input the actual embeddings e_{x_t}, synthetic embeddings e_{z_t}, actual latent vectors h_{x_t}, and the synthetic latent vectors h_{z_t}. The discriminator learns to classify these inputs as real or synthetic, guiding the generator and generator embedder to produce realistic outputs. The adversarial loss L_U is maximized to train the discriminator to improve its ability to distinguish between real and synthetic data.

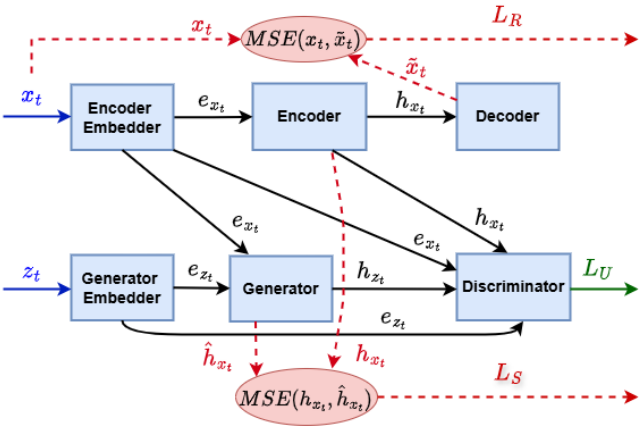


Fig. 3: Diagram of the TimeGAN model.

Let Θ_e, Θ_r, Θ_G, and Θ_D denote the trainable parameters of the embedding network (e: encoder embedder, encoder), recovery network (r: decoder), generator network (G: generator embedder and generator), and discriminator (D) respectively. These components, and their interactions during training, are visually represented in Figure 4, where solid lines denote forward propagation of data, and dashed lines illustrate the backpropagation of gradients. The embedding network (e) and recovery network (r) are both trained on both the reconstruction loss (L_R) and supervised loss (L_S):

$$\min_{\theta_e,\theta_r}\left(\lambda L_S + L_R\right), \quad (5)$$

where $\lambda \geq 0$ is a hyperparameter that balances the two losses. The supervised loss ensures that the embedding process is actively trained to facilitate the generator in learning temporal relationships from the data.

Meanwhile, the generator network and discriminator are trained adversarially, playing an unsupervised Min-Max game over classification accuracy:

$$\min_{\theta_g}\left(\eta L_S + \max_{\theta_d} L_U\right). \quad (6)$$

The discriminator (D) maximizes the adversarial loss by learning to distinguish real data (e_{x_t}, h_{x_t}) from synthetic (e_{z_t}, h_{z_t}). Simultaneously, the generator network (G) is trained to fool the discriminator by generating realistic synthetic data; it also minimizes the supervised loss (L_S) to align synthetic conditional latent representations ($\hat{h}_{x_t}$) with real conditional latent representations (h_{x_t}). The generator's objectives are balanced with a trade-off controlled by the hyperparameter η.

The losses illustrated in Figure 4 reflect the interplay between these components. The autoencoder (embedding network e and recovery network r) optimizes both L_R and L_S to ensure accurate reconstruction and facilitate the generator in learning temporal relations from the data. The discriminator (D) improves its classification capabilities by optimizing its trainable parameters Θ_D to minimize the adversarial loss L_U, to distinguish real and synthetic data. The generator network (G) adapts its parameters Θ_G to fool the discriminator (through L_U) and align with the temporal relationships in the data by optimizing L_S.

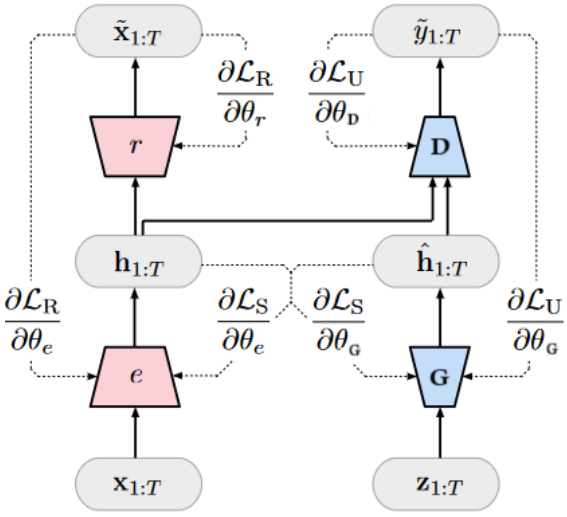


Fig. 4: TimeGAN training scheme, altered from [7].

2.3 *Super-SpaceTime GAN Structure*

This section outlines the modifications made to the TimeGAN structure to adapt it for generating synthetic condition monitoring (CM) data. In Section 2.3.1, the GAN's dual conditioning on the RUL labels and OCs are explained. Following, Section 2.3.2 introduces some necessary alterations to the supervised loss and architecture that ensure GAN adheres to both temporal and spatial dependencies. Finally, Section 2.3.3 presents the proposed Super-SpaceTime GAN structure.

2.3.1 *RUL & OC Conditioning*

We propose a GAN that leverages two key factors for conditioning: the RUL labels and the OCs. This dual conditioning enables the GAN to generate synthetic data that closely reflects the underlying causal relationships between OCs, degradation, and sensor readings.

1) **Conditioning on RUL labels (Z)**

The causal relationship $Z \rightarrow X$ is a critical assumption in prognostics, where the sensor readings (X) capture the performance of the system as it degrades over time. By conditioning the GAN on the RUL labels (Z), the synthetic data reflects the health states of the engines across their lifecycles. This allows the GAN to generate data with a specific degradation state, ensuring that the GAN is steered to generate data that represents various health conditions.

2) **Conditioning on operating conditions (W)**

The causal direction $W \rightarrow X$ captures the influence of the OCs on the sensor readings. For instance, the different flight stages, (e.g. take-off, cruise) lead to distinct sensor readings. By conditioning the GAN on W, the synthetic data incorporates these operational differences. This ensures that the GAN's synthetic CM data incorporates the causal influence of OCs found in the real data.

The causal structure can be formalized using the equations of the structural causal model below [23], where ϵ_1, ϵ_2, ϵ_3 are jointly independent noise variables and f_1, f_2, and f_3 are deterministic causal functions:

$$\text{Operational conditions } W := f_1(\epsilon_1), \tag{7}$$

$$\text{Degradation } Z := f_2(W, \epsilon_2), \tag{8}$$

$$\text{Sensor Readings } X := f_3(W, Z, \epsilon_3. \tag{9}$$

To ensure the conditional information is effectively utilized, the GAN conditions several key architecture components on both the RUL labels and the OCs as depicted in Figure 5. Specifically, we conditioned the encoder embedder and generator embedder such that they incorporate these conditions into the embeddings, enabling the GAN to embed the causal relationships into the latent space. The encoder is not explicitly conditioned, as it primarily trains through the reconstruction loss. However, we explicitly condition the generator and discriminator on the RUL labels and OCs in addition to being indirectly influenced through the embeddings. This dual conditioning reinforces the emphasis on the RUL and OCs during synthetic data generation and classification. The explicit conditioning on multiple levels ensures that the GAN captures the causal dependencies and generates realistic data that aligns with the observed health states and OCs.

Another benefit of the GAN's conditional structure is that it enables the GAN to address data scarcity regions by generating targeted synthetic data. For instance, the GAN can generate data from underrepresented conditions by using the OCs and an initial RUL estimate from a rough model. This data, with RULs in the neighborhood of the estimated values, can be used to augment the training set. Potentially enhancing its prognostic performance when it encounters these initially underrepresented regions during RUL estimation.

2.3.2 *Supervised Spatial TimeGAN*

For the proposed Super SpaceTime GAN, temporal CNNs are used in contrast to the LSTMs used in the original TimeGAN, enabling the GAN to capture both temporal dependencies and spatial relationships between features and conditions. The temporal CNNs are better suited to the conditional structure of the proposed method, allowing the GAN to accurately model the spatial relationships across different features, OCs, and RULs, rather than solely relying on temporal dynamics. We incorporate the conditional structure, denoted as c_t into the Super-SpaceTime GAN. Here, c_t represents the condition at each timestep t, encompassing both the RUL (Z) and OCs (W). Furthermore, to align the GAN with the conditional structure, the supervised- and unsupervised losses have been updated.

In the original setup (Eq. 4), the supervised loss compared the real latent vectors at time t (h_{x_t}) with the synthetic latent vectors at time $t-1$ ($\hat{h}_{x_{t-1}}$). However, this approach is inconsistent with the conditional nature of the proposed method, where OCs can lead to fundamentally different sensor readings. Consequently, comparing the real latent vectors at time $t+1$ with the synthetic latent vectors at time t is inconsistent with the underlying structure and objectives of the model. To address this, the **updated supervised loss**:

$$L_S = \mathbb{E}_{x_{1:T} \sim p} \left[\sum_t \| h_{x_t, c_t} - \hat{h}_{x_t, c_t} \|_2 \right], \tag{10}$$

compares the real latent vectors at time t (h_{x_t, c_t}) with the synthetic latent vectors at time t ($\hat{h}_{x_t, c_t}$). Here, the inclusion of c_t represents the time-dependent conditions (RULs and OCs) embedded with the latent vectors. Incorporating c_t and guaranteeing comparisons at corresponding timesteps, ensures that the generated synthetic data

reflect the spatial dependencies of the real data under identical conditions.

In the original unsupervised (adversarial) loss (Eq. 3), the discriminator evaluated the likelihood of real and synthetic data without explicit consideration of conditional dependencies. To enable the discriminator to classify data samples while taking into account the conditional structure, the **updated unsupervised loss** is defined as follows:

$$L_U = \mathbb{E}_{(x_{1:T},c_{1:T})\sim p}\left[\sum_t \log y_t\right] + \mathbb{E}_{(x_{1:T},c_{1:T})\sim(\hat{p},p)}\left[\sum_t \log(1-\hat{y}_t)\right]. \tag{11}$$

In this equation, y_t and $\hat{y}_t$ represent the discriminator's classification outputs for real and synthetic data, respectively. The notation $(x_{1:T}, c_{1:T}) \sim p$ indicates that the sequences $x_{1:T}$ and their corresponding conditions $c_{1:T}$ are sampled from the real data distribution p. On the other hand, $(x_{1:T}, c_{1:T}) \sim (\hat{p}, p)$ denotes that the sequences $x_{1:T}$ are sampled from the learned data distribution $\hat{p}$, while the corresponding conditions $c_{1:T}$ are sampled from the real data distribution p. In this new formulation, the discriminator evaluates both the data x_t and the conditions c_t to determine whether a sample is real or synthetic. Including c_t ensures that the discriminator stimulates the generator to produce data that aligns with the conditional dependencies defined by the RUL and OCs.

The integration of c_t into both the supervised and unsupervised losses ensures that the Super-SpaceTime GAN generates synthetic data that reflects the temporal- and spatial relationships of the conditional dependencies observed in the real data. These modifications enable the GAN to model complex health degradation patterns under varying operating conditions, significantly improving its utility for CM data augmentation.

2.3.3 *Super-SpaceTime GAN*

The updated diagram in Figure 5 reflects the incorporation of the RUL (Z) and OCs (W) conditions (c_t) into the TimeGAN. As evident from the figure, the encoder embedder and generator embedder are conditioned on the OCs and RULs to produce embeddings that capture these conditions. Additionally, the generator and discriminator are also explicitly conditioned, further reinforcing their influence. This explicit conditioning ensures that the Super-Space GAN models spatial relationships across features and OCs, as well as temporal dependencies concerning health degradation more accurately.

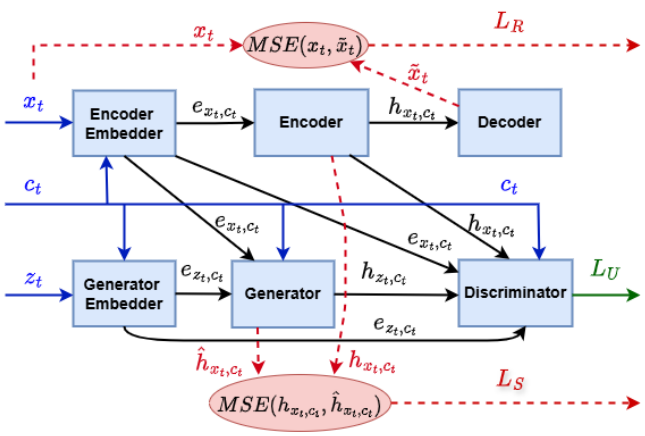


Fig. 5: Diagram of the Super-Space GAN model.

The training hyperparameters of the Super-SpaceTime GAN, depicted in Table I, are tuned to ensure that the synthetic data effectively captures the spatial and temporal dependencies in the data. We implement temporal CNNs with a tuned number of filters, number of layers, kernel sizes, and sequence length. Furthermore, we utilize varying learning rates between the networks and employ a learning rate decay to guarantee smooth convergence. The supervised loss parameters were taken from the original TimeGAN paper, as they reported that the TimeGAN is not sensitive to λ and η [7]. To evaluate the model to allow for tuning, the Jenson-Shannon Distance as explained in Section 2.4, was employed to assess how well the synthetic data aligns with the real data distribution.

TABLE I: Hyperparameter configuration for the Super-SpaceTime GAN model

Model Hyperparameters	
Architecture Parameters	
Neural Network Type	CNN
Number of Filters	96
Number of Layers	8
Kernel Size	10
Sequence Length	40
Training Parameters	
Batch Size	96
Epochs	290
Supervised Loss Function Parameters	
λ (Autoencoder)	1
η (Generator)	10
Learning Rate Configuration	
Autoencoder Initial LR	1.5×10^{-4}
Generator Initial LR	7.0×10^{-5}
Discriminator Initial LR	1.0×10^{-4}
Learning Rate Decay Settings	
Autoencoder Decay Rate	0.90
Generator Decay Rate	0.93
Discriminator Decay Rate	0.97
Decay Steps	4

2.4 *Jenson-shannon Distance*

To effectively tune and monitor the training of a GAN, it is crucial to have a reliable metric for tracking progress. Unlike traditional neural networks, where metrics like RMSE can directly assess prediction accuracy, GANs lack such straightforward measures due to the complexity of distinguishing real from synthetic data. While the generator loss is available, this often fails to reflect the quality of the generated data reliably [24].

To assess this, we employ the *Jensen-Shannon Distance* (JSDist) as a robust metric to evaluate GAN performance and quantify the similarity between real and generated data distributions. The JSDist provides valuable insights into how effectively the GAN captures the underlying data patterns and is thus employed to monitor its performance at each training epoch. Unlike a traditional loss function, the JSDist is a statistical measure of similarity between two probability distributions [25].

JSDist is derived from the Jensen-Shannon Divergence (JSD), a symmetrized and smoothed version of the widely used Kullback-Leibler (KL) divergence. The JSD is defined as:

$$\text{JSD}\left[P(x)\|Q(x)\right] = \frac{1}{2}\text{KL}\left[P(x)\|M(x)\right] + \frac{1}{2}\text{KL}\left[Q(x)\|M(x)\right] \tag{12}$$

where:

$$M(x) = \frac{1}{2}\left[P(x) + Q(x)\right]$$

and KL represents the Kullback-Leibler Divergence, which is defined as follows:

$$\text{KL}\left[P(x)\|Q(x)\right] = \sum_x P(x)\log\frac{P(x)}{Q(x)} \tag{13}$$

where:

- $P(x)$ is the true distribution
- $Q(x)$ is the approximate distribution
- x represents the domain of the random variable.

Since the JSD itself is not a true metric, we compute the Jensen-Shannon Distance by taking the square root of the Jensen-Shannon Divergence:

$$\text{JSDist}(P(x)|Q(x)) = \sqrt{\text{JSD}(P(x)|Q(x))} \tag{14}$$

The JSDist has three desirable properties that make it an excellent fit for quantifying the performance of the synthetic data:

1) **Symmetry**:

$$\text{JSDist}\left[P(x)\|Q(x)\right] = \text{JSDist}\left[Q(x)\|P(x)\right]$$

2) **Boundedness**:

$$0 \leq \text{JSDist}\left[P(x)\|Q(x)\right] \leq 1$$

3) **Numerical Stability**: JSDist is a smoothed version of the KL divergence, making it more stable for distributions where $P(x)$ or $Q(x)$ may have zero probabilities.

The symmetry property of JSDist ensures that the synthetic data distribution $Q(x)$ is modeled after the real data distribution $P(x)$, and at the same time, that $P(x)$ is also made to look like $Q(x)$. Additionally, the boundedness of JSDist ensures that the metric remains meaningful and interpretable, as its values are always constrained between 0 and 1. The smoothness property guarantees the stability of the metric, by accounting for cases where either of the two distributions has zero probabilities, preventing divergence. The combination of symmetry, boundedness, and smoothness makes JSDist an excellent metric for evaluating the quality of synthetic data. Its ability to quantify the alignment between real and synthetic data distributions in a stable and interpretable manner allows for consistent monitoring of the GAN's performance throughout training. Therefore, JSDist is employed as the central tool for tuning the Super-SpaceTime GAN, enabling precise evaluation of the synthetic data quality at each epoch.

3 Case Study - Aircraft Turbofan Engines

The proposed Super-SpaceTime GAN is demonstrated and evaluated in two case studies featuring the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) FD004 dataset. This section explores the CMAPSS set in Section 3.1, and introduces the preprocessing in Section 3.2. The two types of synthetic data used in the data quality and prognostic performance sections are explained in Section 3.3.

3.1 CMAPSS Description

We use the CMAPSS dataset for evaluation in this research [26]. The CMAPSS dataset includes multiple subsets representing different engine degradation scenarios, each containing time-series data from various sensors and operating conditions accompanying these sensor readings. The dataset is divided into training and testing sets, with sensor readings that can be used to predict the RUL of the jet engines.

We use the FD004 subset of the CMAPSS data, the most challenging subset, which employs six different operating conditions and has two failure modes. The FD004 training set consists of 249 units, each representing a complete run-to-failure instance. The lifespans of these training units vary, the shortest lifespan being 128 time cycles and the longest reaching 543 time cycles. The test set includes 248 units, each with varying lifespans. In contrast to the training set, the instances of the test set are not all run-to-failure. The dataset provides the RUL of each unit at its final recorded time cycle, which ranges from 6 to 195 RUL. This structure is designed to simulate a real-world scenario where an engine is monitored until a certain point, after which its RUL must be predicted.

Figure 6 showcases the six groups of OCs found in the CMAPSS FD004 dataset.

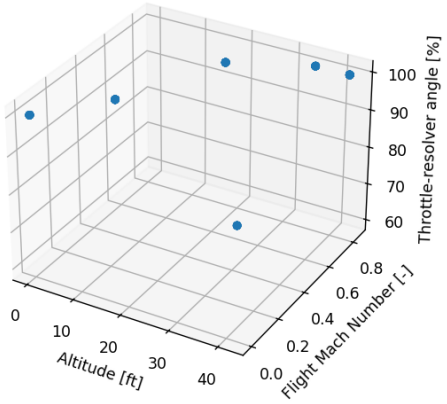


Fig. 6: The six operational conditions of the CMAPSS FD004 dataset.

3.2 Pre-processing

The CMAPSS FD004 dataset employs 21 sensors, from which 7 have a constant value over time and are thus omitted, the remaining 14 sensors will be used for their sensor measurements. Furthermore, the data preprocessing steps include normalizing the sensor measurements using min-max normalization w.r.t. the operating conditions [27], [28]:

$$\hat{m}_{ij} = \frac{2\left(m_{ij}^{k} - m_{jk}^{\min}\right)}{m_{jk}^{\max} - m_{jk}^{\min}} - 1 \tag{15}$$

where m_{ij}^{k} is the sensor measurement of sensor j during flight i under the operating conditions k, $m_{jk}^{\max}$ and $m_{jk}^{\min}$ respectively are the maximum and minimum sensor measurement of sensor j under operating condition k. Finally, $\hat{m}_{ij}$ is the normalized sensor measurement of sensor j during flight i,

To prioritize the GAN's and RUL model's focus on the degraded portion of the sensor measurements and avoid overemphasis on negligible differences in healthy data, we employ a piece-wise linear RUL target function [29], [30]. Specifically, the RUL labels are capped at 125 flights, such that for data points with $RUL \geq 125$ the target value is set at $RUL = 125$ flights.

3.3 Types of Synthetic Data

We use the Super-SpaceTime GAN to generate two types of synthetic data, which are fed through the GAN in sequences of length 40:

- **Synthesized real data**

This type of data is generated by taking the exact OC and RUL labels of the real training/test data. The GAN takes these exact real OC and RUL labels and generates synthetic sensor readings that it deems fit to the real conditions inputted. These synthesized real data are crucial for allowing proper data quality evaluation, as they allow comparison of synthetic sensor readings with real sensor readings under the same conditions.

- **Fully synthetic data**

For this synthetic data type, the OC labels are sampled from the distribution of OCs in the real CM data. While the frequency distribution of OCs is preserved, the sequence of occurrences is randomly sampled, making it independent of the real data. Furthermore, all of the fully synthetic sequences belong to a unit that is a run-to-failure instance; the last time cycle of each unit is therefore equal to zero. To emphasize the degraded portion of the training data, the maximum time cycle for each unit is drawn from a uniform distribution between 40 and 200 time cycles, resulting in a mean life cycle duration of 120 time cycles. For instance, generating 10,000 fully synthetic sequences corresponds to approximately 83 run-to-failure units. This type of data is essential for augmentation purposes as augmenting data sets

with synthetic data tied to the exact OC and RUL labels of the real data is ineffective for increasing prognostic performance.

We solely use synthesized real training and test data for the synthetic data quality evaluation experiments in Section 4. For the prognostic performance evaluation, in Section 5, except explicitly mentioned, we use fully synthetic data to augment the existing real data set.

4 Data Quality - Case Study

In this section, we evaluate the data quality of the synthetic data generated by the Super-SpaceTime GAN through both quantitative and qualitative inspection. The experiments aim to assess how closely the synthetic aligns with the real data in capturing sensor measurement dynamics and modeling health degradation patterns. To achieve this, we utilize the Jenson-Shannon Distance, t-distributed Stochastic Neighbor Embedding (t-SNE), and Principal Component Analysis (PCA). Additionally, we use a novel autoencoder-based residual method to evaluate the alignment of health degradation patterns between the synthetic and real data. The experiments, as outlined in Section 4.1, are conducted on the FD004 dataset described in Section 3. This analysis, depicted in Section 4.2, provides an in-depth investigation of the GAN's performance on both inside-of-distribution from the training set and outside-of-distribution from the test set. By comparing the training- and test-set scenarios, we gain valuable insights into the GAN's strengths and weaknesses in handling known and unseen data. The results of the in-depth evaluation of the Super-SpaceTime GAN synthetic data quality provide recommendations for improving its capabilities, particularly in addressing issues like overfitting specific OC-RUL patterns and mode collapse. Finally, the limitations found in the experiments are summarized in Section 4.2.4.

4.1 Evaluation Metrics

In this section, we explore the evaluation metrics used to evaluate the data quality of the Super-SpaceTime GAN's synthetic data. We analyze the data quality during training by monitoring the Jenson-Shannon Distance in Section 4.1.1. Then, Section 4.1.2 shows the visual data inspection, where we use PCA for global data visualization and t-SNE for more localized inspection. Lastly, Section 4.1.3 introduces a novel autoencoder-based metric that assesses the health degradation patterns in the synthetic data.

4.1.1 Jenson-Shannon Distance

We utilize the Jensen-Shannon Distance (JSDist), as explained in Section 2.4, to evaluate the quality of synthetic data generated by the Super-SpaceTime GAN during training. By measuring the distance between real and synthetic data distributions, JSDist provides a quantitative metric to assess how closely the GAN aligns the synthetic data with the real data. Unlike generator loss, which may not reliably indicate data quality, JSDist offers a stable and interpretable measure to monitor progress during training and evaluate the data quality at each epoch. Lower JSDist values indicate better alignment, making it a key metric for assessing and tuning the GAN's performance, especially during training.

4.1.2 t-SNE & PCA for Visualizing Data Distributions

For visual evaluation of the synthetic data, we use t-distributed Stochastic Neighbor Embedding (t-SNE) and Principal Component Analysis (PCA), which are both widely used dimensionality reduction techniques.

t-SNE: t-Distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear dimensionality reduction technique that visualizes high-dimensional data in a lower dimensional space [31]. It works by modeling pairwise similarities found between the data points in the high dimensionality and preserving those similarities in the lower-dimensional embedding. It employs a probabilistic approach where it aims to minimize the Kullback-Leibler distance (Eq. 13) between a Gaussian distribution in the higher-dimensional space and a t-distributed distribution in the lower-dimensional space. Its ability to reveal local groupings of data and structures makes it a valuable tool for assessing the similarity between real and synthetic data.

PCA: Principal Component Analysis (PCA) is a linear dimensionality reduction technique that transforms the data into a set of linear uncorrelated variables called principal components. Following, these components are ordered by the amount of variance they explain, where the first principal component shows the most variance. It determines the direction of maximum variance by calculating the eigenvectors and eigenvalues of the covariance matrix of the data. In contrast to t-SNE, PCA preserves the global structure of the data, making it an excellent tool for analyzing overall trends and patterns between the synthetic and real data.

By projecting data into a lower-dimensional space, t-SNE and PCA allow for a clear visual assessment of the similarity between distributions. In this report, we use t-SNE and PCA to evaluate how closely the synthetic data distributions align with the underlying real data distributions, providing a clear visual representation of their similarities and dissimilarities.

4.1.3 Residual Method for Monitoring Health Degredation

Health degradation patterns in sensor readings are critical indicators for assessing whether the GAN's synthetic condition monitoring data (CM) properly mimics the underlying real data dynamics. For synthetic CM data to be beneficial to a RUL model to learn meaningful degradation patterns, it is paramount that the health degradation patterns in the synthetic CM data accurately reflect the health degradation shown in the real data. A widely adopted method to quantify health degradation in CM data is the autoencoder-based residual method [32]–[36]. The AE-based residual method uses CM data labeled as healthy, to train an AE model to accurately encode and reconstruct the healthy CM data X into the reconstructed CM data $\hat{X}$. This healthy data is sourced from the initial operating cycles of each engine, where RUL labels (¿125) confirm their healthy condition. Once the AE is trained to reconstruct healthy CM data accurately, it is then used to generate the reconstruction residual r of the current CM data X, measuring the deviation between the original and reconstructed signals:

$$r = |\hat{X} - X|. \tag{16}$$

The residual r, originally spanning 14 sensor dimensions, goes through a dimensionality reduction by averaging across timesteps and features. This is followed by a min-max normalization to obtain the health index (HI), which is a normalized one-dimensional projection in the range $[0, 1]$:

$$r \in \mathbb{R}^p \rightarrow h \in \mathbb{R}. \tag{17}$$

The residual method is based on the principle that higher reconstruction residuals indicate deviations in the current CM data that the autoencoder (AE) has not been trained to encode or reconstruct. Since the AE is trained exclusively on healthy data, difficulties arise when the AE attempts to encode and reconstruct faulty data. As a result, the AE model struggles to emulate faulty CM data as it is unfamiliar with the underlying degradation dynamics; leading to greater residuals. These reconstruction residuals, therefore, are an outstanding measure for estimating the health index in CM data. The greater the health degradation, the more significant the AE's inability to accurately emulate the current CM data becomes. The autoencoder-based residual method is visually depicted in Figure 7. In the figure, the CM data inputted into the AE consists of the sensor readings X and the operating conditions W, leading to the reconstructed $\hat{X}$, this therefore considers the causality between X and W.

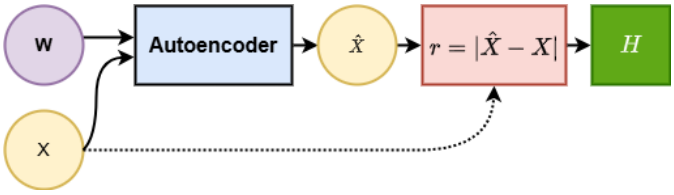


Fig. 7: Autoencoder-based residual method for health index estimation, based on [23].

The AE-based residual for health index estimation can be summarized in the following steps:

1) **Train the Autoencoder (AE):**

First, we use healthy real CM data as input to train the AE. The AE consists of an encoder with three dense layers of 64, 32, and 16 units, concatenating sensor measurements and operating conditions (OCs). The decoder mirrors the encoder with three dense layers of 16, 32, and 64 units and is designed to reconstruct the sensor measurements.

2) **Calculate Residuals:**

After training, we pass healthy and faulty real CM data through the trained AE to compute reconstruction residuals. Here the residuals for synthetic healthy and faulty CM data are computed with the same trained AE.

3) **Dimensionality Reduction and Normalization:**

Following, we reduce the dimensionality of the residuals by averaging across timesteps and features. After which min-max normalization to obtain a one-dimensional health index (HI) in the range $[0, 1]$ is performed.

4) **Compare Health Indices:**

Finally, we analyze the normalized 1-dimensional HIs to assess the similarity between the HIs of real and synthetic healthy data. We compare the HI trajectories of real and synthetic faulty CM data to confirm that the synthetic data captures the same exponential health degradation patterns as the real data.

4.2 Data Quality Evaluation Results

This section presents the results of the data quality evaluation experiments. First, Section 4.2.1 showcases the training progress of the JSDist. Section 4.2.2 shows the visual data inspection graphs (PCA and t-SNE). In Section 4.2.3, the results of the health degradation monitoring via residual method are presented.

4.2.1 Jenson-shannon Distance

By employing the Super-Space GAN with the hyperparameters described in Section 2.3.3, optimized to minimize the JSDist as possible for the GAN trained on the full dataset (249 units), we obtain the training progress depicted in Figure 8. In this figure, the JSDist of the test set is depicted for a GAN trained on the full dataset of 249 units (depicted in red) and a GAN trained on a limited dataset of 50 randomly generated units (depicted in blue).

These two training sets were used to evaluate the GAN under different data availability circumstances. In the first scenario, the GAN is trained on 50 units and tested on the whole test set of 232 units. This setup assesses the performance of the GAN in a data-scarce environment, where data augmentation is expected to be the most beneficial and desirable. In the second scenario, the GAN is trained on the full training set of 249 units and tested on the same test set of 232 units, serving to assess the GAN's performance under data-rich conditions.

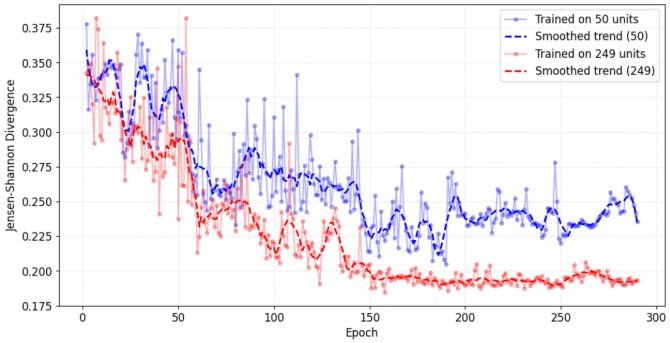


Fig. 8: Raw and smoothed Jensen-Shannon Distance across epochs for the GAN trained on 50 and 249 units.

The graph shows that the JSDist of the GAN trained on the full dataset is lower than that of the GAN trained on the limited dataset, meaning that the synthetic data generated by the GAN trained on the full dataset aligns more closely with the underlying real data than that of the GAN trained on the limited dataset. Furthermore, the JSDist of the GAN trained on the limited dataset of 50 units shows greater variability in JSDist across epochs than the GAN that has access to the full training set.

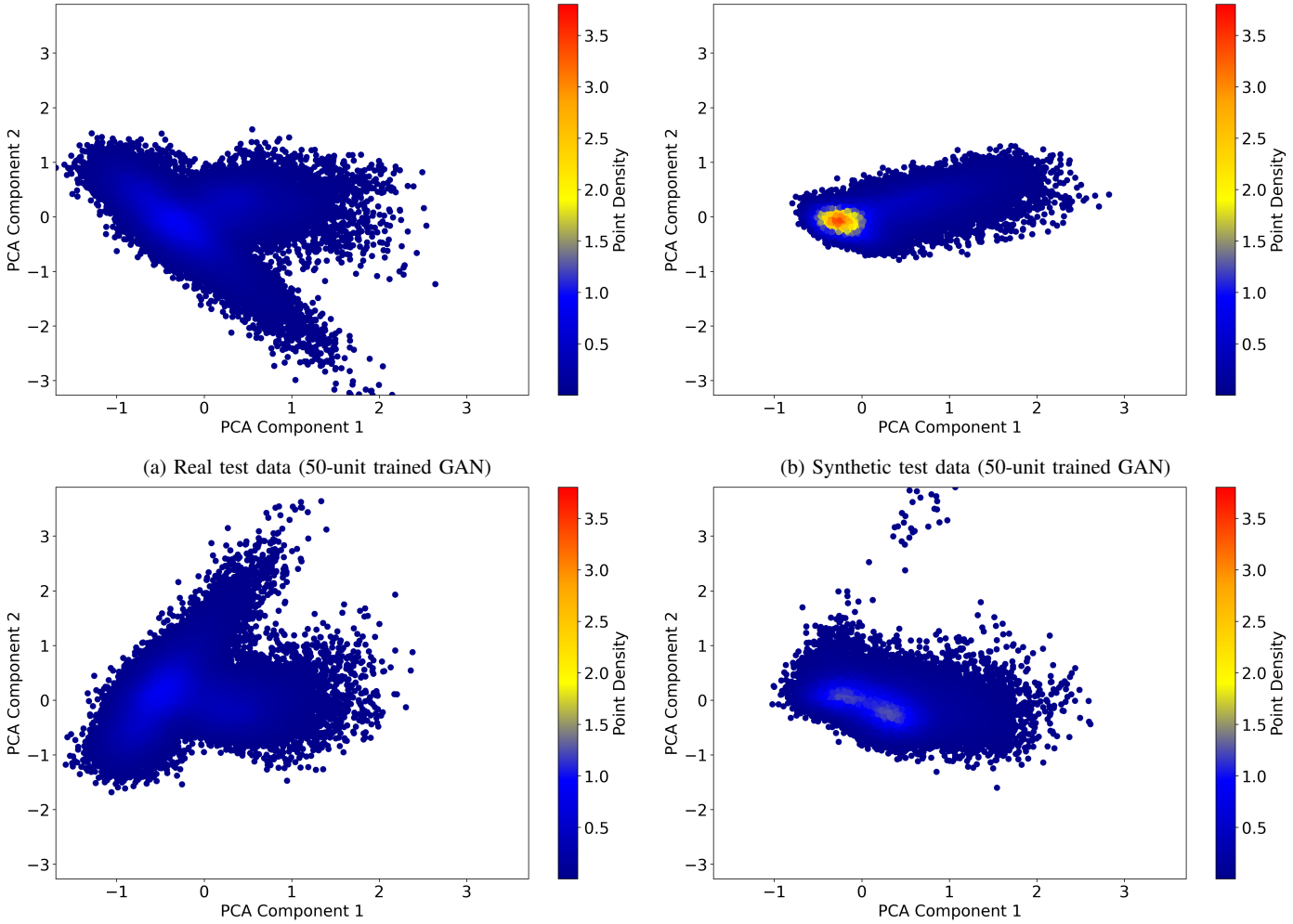
4.2.2 Visual Data Inspection

Principal Component Analysis

The graphs in Figure 9 provide a comparative analysis of the PCA visualizations for real and synthetic data distributions within the test set. We have four plots showing the first two principal components, with point density indicated by color intensity from blue to red. Figure 9a and Figure 9c represent the real data's PCA structure when performing PCA reduction paired with a synthetic dataset generated by a GAN trained on a limited dataset of 50 units and the full dataset of 249 units respectively. While the real test data itself remains the same between the two graphs (Figure 9a and Figure 9c), the PCA identifies directions of maximum variance in the combined dataset (real + synthetic) and thus influences the PCA reduction process, giving inherently differing results when performed with different synthetic data. In both figures, two distinct PCA strands can be observed in the real data. These strands likely correspond to the two failure modes present in FD004, as the dominant variance in sensor readings over time is driven by health degradation patterns associated with failure modes [37].

The synthetic data generated by the 50-unit GAN, illustrated in Figure 9b, exhibits significant limitations in capturing the full dynamics of the underlying real data distribution. The synthetic data forms a highly concentrated region around the origin (0,0), with a red-yellow hotspot. This concentration, absent in the real data (Figure 9a), strongly suggests that the 50-unit GAN suffered from mode collapse. Mode collapse is a phenomenon where the GAN fails to generalize the full data distribution but rather focuses on generating samples from a small subset of the distribution. Moreover, the 50-unit GAN captures only a single strand (failure mode) of real data's PCA structure, failing to represent the full range of variability found in the real data. In this case, the mode collapse is likely caused by the GAN generating synthetic data focused on one failure mode only. Even within that data strand, the synthetic data lacks the breadth and variability of the underlying real data. This narrow representation and limited data variability underscore the lack of capacity to generalize the full real data distribution for the GAN trained on a limited dataset.

In contrast, as depicted in Figure 9d, the synthetic data generated by the GAN trained on the full dataset (249 units) has a broader spread and improved overlap with the real data distribution. The synthetic data has better alignment with the real data distribution (Figure 9c). When examining the distribution coverage, the 249-unit GAN effectively captures the main strand of the real data's PCA structure and even shows some minor overlap with the secondary



(a) Real test data (50-unit trained GAN)

(b) Synthetic test data (50-unit trained GAN)

(c) Real test data (249-unit trained GAN)

(d) Synthetic test data (249-unit trained GAN)

Fig. 9: PCA visualization of real vs synthetic test data. Subfigures (a) and (b) depict PCA results for real and synthetic data generated by a GAN trained on a limited dataset (50 units), while subfigures (c) and (d) show PCA results for real and synthetic data generated by a GAN trained on the full dataset (249 units).

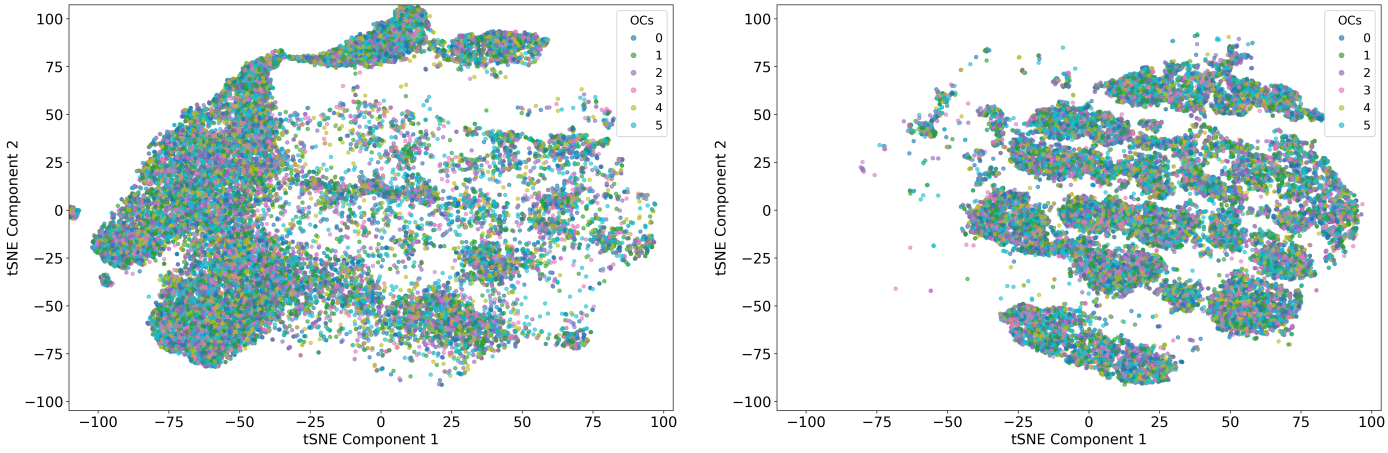
strand. Additionally, the broader and more accurate representation of the primary strand of the real data shows the improved ability to model intra-strand variability. This partial coverage indicates that the 249-unit GAN begins to generalize beyond a single dominant feature set, offering a more diverse representation of the real data. Despite this improvement, the 249-unit GAN still exhibits a slightly more concentrated density hotspot at the origin compared to the real data. Furthermore, the 249-unit GAN struggles to effectively capture the second data strand, suggesting that it struggles to model one of the failure modes. This indicates that while the GAN trained on the full datasets mitigates some of the mode collapse, it does not eliminate the issue.

Overall, the 249-unit GAN shows smoother transitions in density and captures the shape and density variations of the real data better than the limited data GAN. However, even for the GAN trained on the full dataset it shows a form of mode-collapse, causing one of the two failure modes to be underrepresented.

t-Distributed Stochastic Neighbor embedding

In, Figure 10 the real and synthetic data of the test set generated by the GAN trained on the full dataset is depicted. Each of the six OCs found in the FD004 dataset has a different colored dot in the graph. There are several groups and structures of data found in the real data distribution depicted in Figure 10a, and it can be noted that the OCs do not form clusters and that they are evenly spread over the local groups found. The distribution of OC labels of the synthetic test data's t-SNE embeddings found in Figure 10b cover the entire graph and are spaced evenly (not concentrated) in each region. This indicates that the synthetic test data generated by the GAN is not biased toward creating clusters of data for OCs. The GAN effectively generates a broad range of variability for each of the OCs and does not suffer from mode collapse concerning OCs.

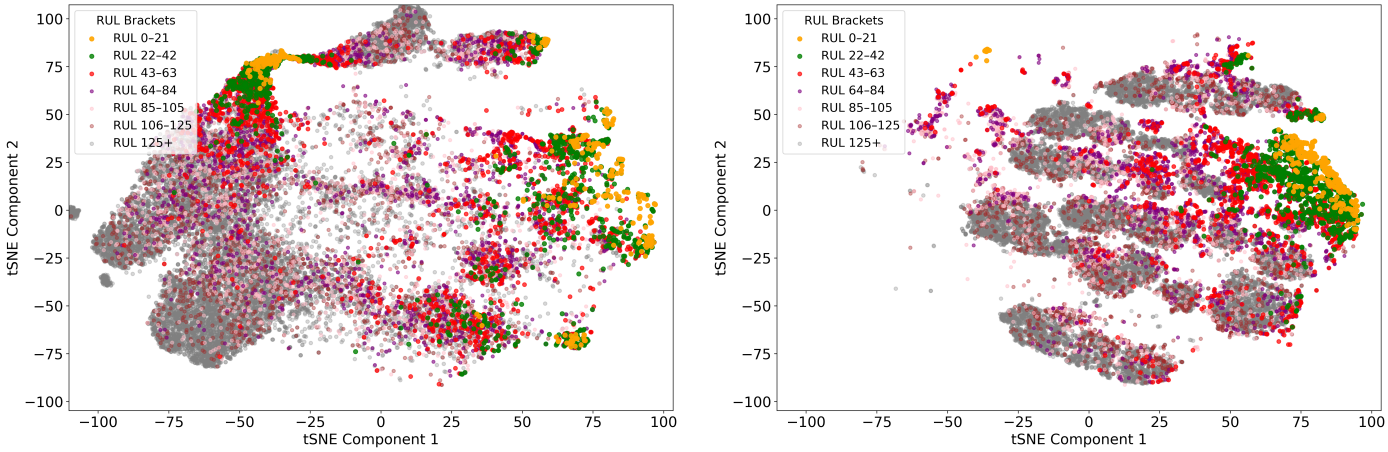
To examine how synthetic data with different RULs is distributed, we refer to Figure 11. The RUL bracket for each sequence (length 40) is determined based on the RUL value of the last timestep in that sequence (i.e. the lowest RUL). Comparing the real and synthetic distributions, we see that the lower RUL brackets in the synthetic data (Figure 11b) appear to be much more tightly clustered than the real data (Figure 11a). In particular, RUL values 0-21 (orange) and 22-42 (green) in the synthetic data are clustered predominantly in the middle-right part of the t-SNE space, whereas in the real data, they are spread across multiple regions. While the lower RUL values in the synthetic data exhibit some dispersed points in the top-left and top-right, these are sparse and form no significant clusters. This aligns with the PCA analysis, suggesting that the GAN may have overfit to mostly a single dominant health-degradation trajectory for the lower RUL values. Effectively, the GAN shows a form of mode collapse for the lower RUL classes. However, for mid-to-high RUL brackets (e.g. 43-63, 64-84, and 84+), the synthetic distribution shows a broader coverage of RULs that matches the real data's structure



(a) Real training data

(b) Synthetic test data

Fig. 10: t-SNE of Real vs Synthetic Data generated by the 249-unit GAN, with OCs highlighted.



(a) Real training data

(b) Synthetic test data

Fig. 11: t-SNE of Real vs Synthetic Data generated by the 249-unit GAN, with RULs highlighted.

more closely. This indicates that the GAN captured a diverse range for early- to mid-life engine degradation trajectories, but overfitted to certain degradation patterns for engines close to failure.

4.2.3 Residual Method for HI Estimation

The residuals visualized in Figure 12 illustrate the residual patterns of both the real data and synthetic data, providing valuable insights into the GANs ability to model the health degradation of failing units effectively. The units selected for the graphs in Figure 12 serve the purpose of showing one synthetic unit that closely aligns with the real data (unit 31 of training set and unit 135 of test set), one that shows average performance (unit 56 of training set and unit 12 of test set), and finally one that does not align with the real data (unit 184 of training set and unit 31 of test set).

Note that the GAN generates independent sequences of length 40 without leveraging context from preceding or succeeding sequences, that is it only uses the context from the current sequence it is working on. Combined with the input random noise, this approach results in less smooth health degradation for the synthetic data compared to the real data, for which the sequences are interdependent. This does not impact the utility of the synthetic data for RUL model data augmentation, as typical RUL models do not rely on sequence interdependencies. Consequently, while the health degradation of the synthetic data may appear less smooth, this does not undermine the quality of the data for typical data augmentation purposes. However, for synthetic data to be effective for data augmentation, the overall health degradation trend across units should exhibit a monotonically increasing exponential pattern.

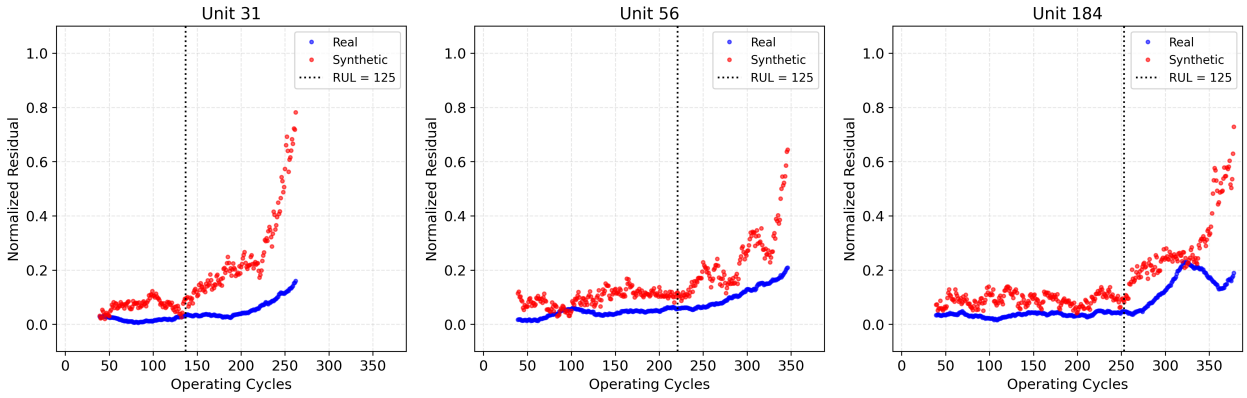
50-unit GAN: synthetic training data

The synthetic data generated by the 50-unit GAN overestimates health degradation compared to real data across all three units consistently (Figure 12a). For all three units, the synthetic data residuals exhibit large oscillations in the non-degraded phase (RUL < 125), showing a significant oscillation in the health degradation where the real data does not. Toward RUL 125, at the dotted line, the synthetic data display excessive health degradations, deviating significantly from the exponential trend of the real data, both in steepness and shape. Rather than the exponential shape of the real data, the synthetic data follows a steep linear trajectory punctuated by sharp oscillations. These shortcomings indicate overfitting to specific patterns within the limited training data. This mode collapse causes the GAN to generate synthetic data that exaggerate health degradation trends.

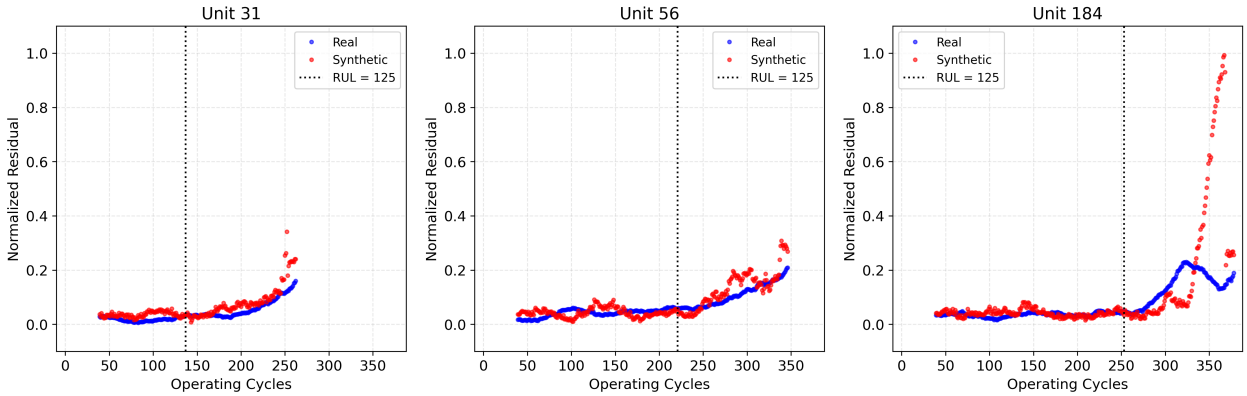
The results for the 50-unit GAN's synthetic test data are very similar to the 50-unit GAN's synthetic training data but show even greater overestimated variability. The plots for the 50-unit GAN for synthetic test data are omitted from this paper for brevity.

249-Unit GAN: synthetic training data

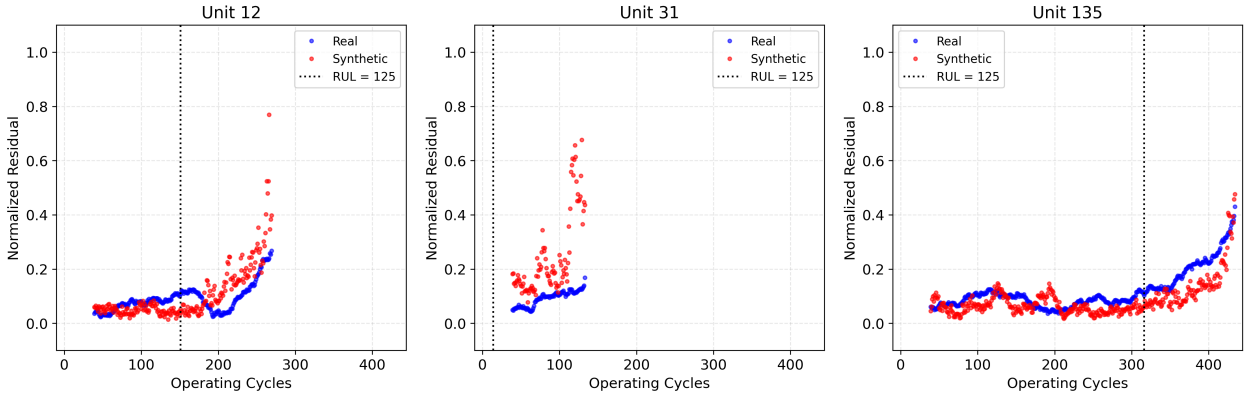
The 249-unit GAN shows significant improvements over the 50-unit GAN, as is evident from Figure 12b. For all units (31, 56, and 184) the synthetic data's healthy phase closely matches the real data, with oscillations in that region remaining within feasible bounds. These oscillations, although not perfectly aligned with the real



(a) Real training data and synthetic training data generated by a GAN trained on a limited data set of 50 units.



(b) Real training data and synthetic training data generated by a GAN trained on the full data set of 249 units.



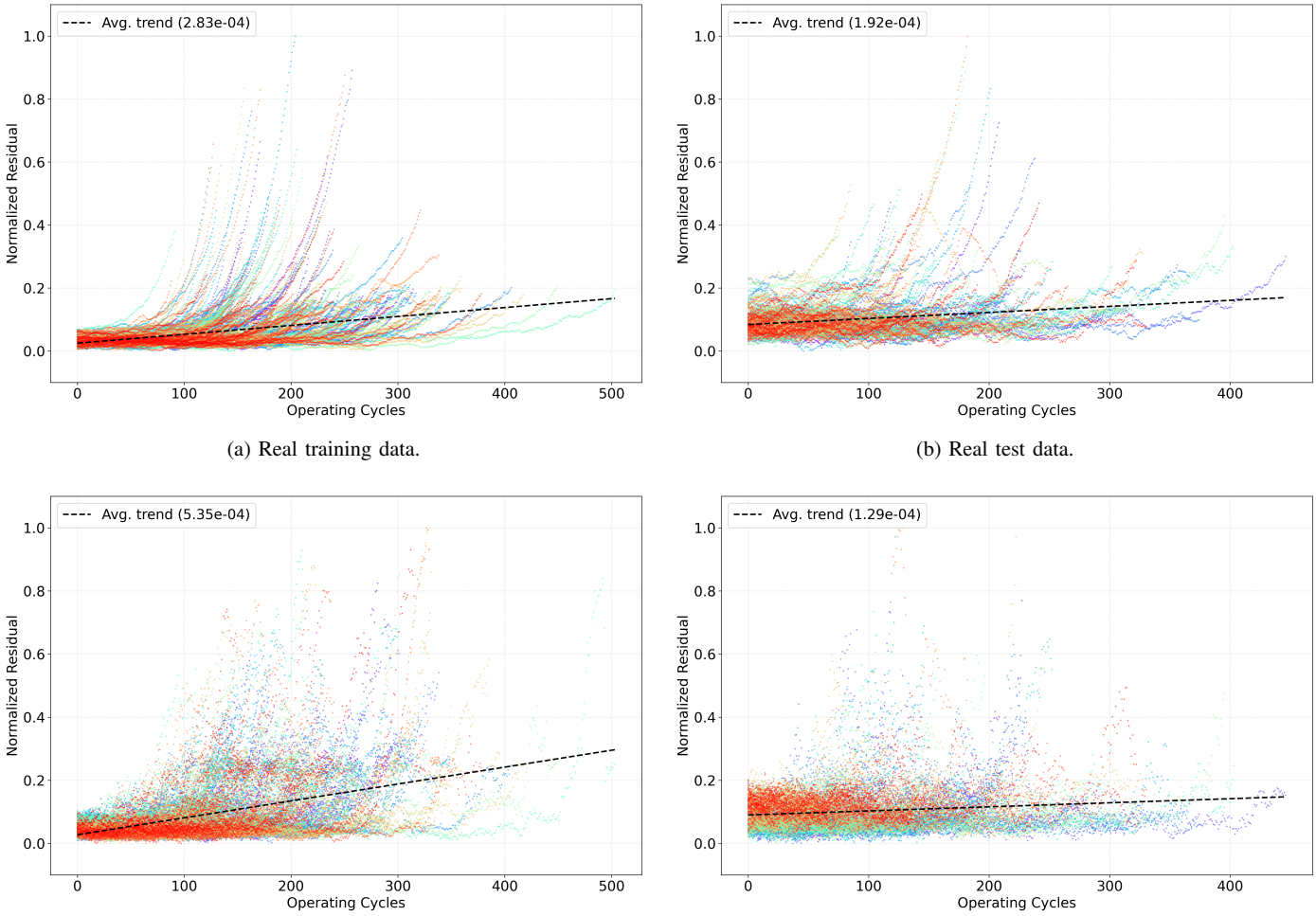
(c) Real test data and synthetic test data generated by a GAN trained on the full data set of 249 units.

Fig. 12: Estimated health residuals of the real- and synthetic training and test data across GANs trained with 50 and 249 units.

data, are acceptable given the GAN's random noise input and the independence of generated sequences. This variability does not diminish the quality of the synthetic data, as the health degradation trends are preserved. In the degraded phases, the synthetic data for unit 31 follows an exponential health degradation trend that closely aligns with the real data. However, during the final 15 cycles, a slight overestimation of the health degradation arises. For unit 56, the synthetic data maintains a reasonable similarity to the real data until the last 100 cycles, where oscillations grow larger and the final degradation is again slightly overestimated. For unit 184, once the final 100 cycles are reached, the health degradation increases with an extremely steep slope. The last 10 cycles, however, show an abrupt decrease in the residuals, almost even reaching the expected levels of degradation of real data. Overall, the synthetic data tends to overestimate the health degradation for the last 10-20 cycles, after which it stabilizes somewhat, settling at a slightly overestimated level of degradation in the cycles immediately preceding failure.

249-Unit GAN: synthetic test data

In Figure 12c, the GAN generated synthetic test data for combinations of OCs with RUL sequences it has never seen before, indicating how well the GAN generalizes to unseen conditions. Across unit 12 and unit 125 (with final RULs equal to 7), the synthetic data's healthy phases align well with the real data. The oscillations in the residuals are within a feasible range, demonstrating that the GAN can generalize healthy dynamics to unseen conditions. While the oscillations are not perfectly in phase with the real data, this is not a concern, as multiple degradation patterns may exist for a given OC-RUL combination. The feasibility and boundedness of the synthetic data's oscillations are more of interest, as they ensure realistic variability. For unit 135 the degradation closely follows the exponential degradation of the real data, demonstrating the GAN's ability to successfully generalize the degraded health dynamics to unseen conditions for this case. For unit 12, the synthetic data matches the overall degradation pattern



(a) Real training data.

(b) Real test data.

(c) 249-Unit GAN's synthetic training data.

(d) 249-Unit GAN's synthetic test data.

Fig. 13: Comparison of the aggregated estimated health indexes of the real and synthetic train and test.

of the real data, but once more, the final 10 cycles are significantly overestimated. Thirdly, unit 31 presents a clear failure, the faulty data shows highly overestimated health degradation throughout and contains an unrealistically severe oscillation. Unit 31 suggests that the GAN struggles with certain OC-RUL combinations not encountered during training.

Aggregated Residual Analysis

The aggregated residuals depicted in Figure 13 show a broader perspective of the health degradation across units by combining the data from all the units. Additionally, the average health degradation across all units is included, giving a valuable comparison between the average health degradation trends of real data and synthetic data. Note that the real training data (Figure 13a) shows a higher average health degradation trend compared to the real test data (Figure 13b). This difference arises because many units in the test data set are not till failure, resulting in lower final degradation levels.

The synthetic data depicted in Figure 13c exhibits less smoothness than the real training data. Furthermore, the synthetic data exhibit significantly higher average degradation trends than the real training data. This is consistent with the overestimation of health degradation seen in the individual unit plots and confirms that the GAN tends to generate data with an exaggerated health degradation for the training set. Observing the degradation trends in the synthetic data reveals that the GAN predominantly generates data with a pronounced exponential degradation pattern, similar to those found in the earlier regions of the real data in Figure 13a. The GAN appears to neglect the units with a more gradual, elongated degradation pattern, such as units with higher lifespan (> 300) in the real data. The GAN opts for the faster, more abrupt exponential degradation pattern, even in cases where typically a slower degradation trend is expected based on the real data.

In the synthetic test data (Figure 13d), the health degradation trajectory lacks smoothness, which is caused by the independent nature of the sequences generated by the GAN. Despite this, the average health degradation trend in the synthetic training data is noticeably lower than in the real training data. This discrepancy appears to stem from the GAN's difficulty in generating realistic failure data (RUL $<$ 20) for unseen OC patterns. For low RUL values in the test set scenarios, the GAN seems to overfit to generating data with relatively low health degradation. This issue is consistent with observations from the t-SNE plot (Figure 11b), which reveals mode collapse for synthetic test sequences with RUL < 40. The GAN predominantly generates failure data with minimal health degradation, while for slightly higher RUL values (20–40), the health degradation is more pronounced. This counterintuitive behavior results in a negative health degradation trend as the final degradation values are lower than those for the slightly higher RUL sequences, reducing the overall average health degradation trend. This explains why the synthetic test data exhibits a lower average trend compared to the real test data.

The significantly higher average health degradation trend in synthetic training data indicates that the GAN overestimates health degradation for known OC-RUL combinations, particularly in the final cycles before failure. Conversely, the lower average trend in test data reveals that the GAN underestimates health degradation for unknown OC-RUL combinations, particularly for low RUL values. This divergence provides strong evidence that the GAN has overfitted to the specific OC-RUL patterns present in the training data for failure scenarios.

4.2.4 Challenges & Limitations

In this section, we address the main challenges and limitations encountered during the data quality evaluation of the proposed Super-SpaceTime GAN. Regarding the data quality of the Super-SpaceTime GAN's synthetic data, we found three limitations:

1) **Mode collapse:** The GAN exhibited a tendency to overfit to dominant failure modes, limiting generalization capabilities.
2) **Exaggerated degradation patterns:** The GAN often generated overly steep degradation patterns.
3) **Dependence on OC-RUL combinations:** In some cases, the GAN struggled to generalize across diverse OC-RUL combinations, particularly for highly degraded phases.

Addressing these three limitations is critical to enhancing the practicality of the Super-SpaceTime GAN for RUL model prognostic performance enhancement. This will especially aid in generating synthetic data that captures a broader spectrum of degradation patterns, increasing the prognostic performance enhancement of the data.

5 Prognostic Performance - Case Study

This section evaluates the prognostic performance of the synthetic data generated by the proposed GAN framework. The primary objective of the synthetic data generation is to augment an existing dataset with synthetic CM data that closely mimics the real CM data distribution. This augmentation aims to create larger, more diverse datasets, potentially enhancing the performance of RUL models. While previous evaluations indicate that the synthetic data distribution aligns closely with the underlying real data distribution, the critical question remains: can this synthetic data effectively improve the performance of RUL models? To address this, we conduct a series of experiments to assess whether augmenting datasets with the synthetic data results in improved RUL model performance compared to models trained on the original, non-augmented dataset.

This section is structured as follows. Section 5.1 provides an overview of the CNN-based RUL prediction model used for evaluating the synthetic data. In Section 5.2, the experimental setup used to assess the utility of the synthetic data for prognostic enhancement is outlined. The results of the prognostic performance evaluation are depicted and discussed in Section 5.3. Finally, Section 5.3.4 addresses the challenges encountered during the prognostic evaluation of the Super-SpaceTime GAN's synthetic data augmentation experiments.

5.1 Remaining Useful Life Model

The Remaining Useful Life (RUL) prediction model is an integral component for the prognostic evaluation of the synthetic data. We utilize a CNN-based architecture to predict RUL based on sensor readings, which is drawn upon the RUL model implementation developed by de Pater et al. (2022) [28]. The CNN layers process multi-dimensional data samples X, representing the sensor measurements corresponding to the operating conditions of an engine over time. For a specific unit i, the input sample x_i is defined as:

$$x_i = \left[\hat{m}_i^1, \hat{m}_i^2, \ldots, \hat{m}_i^M\right]$$

where $\hat{m}_i^j$ is the normalized measurement of the j-th sensor during flight i and M is the total number of sensors.

The CNN architecture, depicted in Figure 14, includes $L = 5$ convolutional layers, a fully connected layer, and a linear output layer:

- The first four convolutional layers each contain $K_f = 10$ kernels of size $K_s = 10 \times 1$.
- The fifth, last convolutional layer, has a single kernel of size $K_s' = 3 \times 1$, combining the feature maps from previous layers into a single representation.
- All convolutional layers use *same padding* to ensure a constant dimension across the feature maps and use the hyperbolic tangent (tanh) activation function.
- The fully connected layer, containing 100 neurons applying the tanh activation function, flattens the feature map into a 2D representation and includes a dropout layer to reduce overfitting during training.
- The Final output layer consists of a single neuron with a linear activation function to predict the RUL.

The network is optimized using the Adam optimizer with an initial learning rate of 0.001, which reduces by a factor of 0.6 after 10 consecutive epochs without improvement. Training is performed over 150 epochs using a batch size of 256 samples with sequences of length 19. Root Mean Square Error (RMSE) is used as a loss function for the network:

$$RMSE = \sqrt{\frac{1}{n}\sum_{w=1}^{n}\left(RUL_w^{actual} - RUL_w^{predicted}\right)^2} \tag{18}$$

The same RMSE metric is consistently used to evaluate the RUL model's performance on the test set across the different experiments. In each experiment, the RMSE is calculated at the final timecycle of each unit. Since all experiments in this section are performed on the test set, the RMSE is calculated at the last available time cycle for each unit, where the RUL varies as the units in the test set do not progress to actual failure.

5.2 Cases

In the RUL model testing phase, we evaluate the prognostic performance of the synthetic data under various conditions, involving non-augmented training sets, augmented training sets, and even training sets where the data is fully replaced with synthetic data. Specifically, the RUL model's prognostic performance is analyzed in the cases highlighted in Table II, as described below:

Replacement (Experiment 1): In this experiment, we investigate the performance of the RUL model when all real training data is replaced with two types of synthetic data. We compare a RUL model trained on fully synthetic data with one trained on synthesized real data, without any real data. We assess whether the fully synthetic data achieves comparable prognostic performance or whether the GAN overfits on the training data and its operating condition (OC) patterns. This experiment effectively isolates the influence of the training data by eliminating dynamics with the original real training data. As a result, it provides a direct head-to-head comparison between synthetic data generated by a GAN trained on 50 units and a GAN trained on the full dataset.

Augmentation (Experiment 2): In this setup, we train the RUL model on reduced datasets augmented with fully synthetic data. The synthetic data is generated using two GAN configurations: one trained on the full FD004 dataset (249 units) and one trained on a reduced dataset of 50 units. Augmentation amounts of 1,000, 2k, 5k, 10k, and 20k sequences are added to the reduced dataset. This experiment evaluates the effectiveness of different augmentation configurations in enhancing the RUL model's performance, especially as the training size decreases. Furthermore, it investigates how the prognostic performance of synthetic data generated by a GAN trained on a smaller dataset compares to that of a GAN trained on the full dataset for data augmentation. Ultimately, this experiment aims to identify the best augmentation configuration for achieving the most significant improvement in prognostic performance.

Baseline vs. Augmented (Experiment 3): This final experiment serves to compare the best data augmented setup versus the benchmark which is training the RUL model on the complete FD004 dataset (249 units) and progressively reducing the training

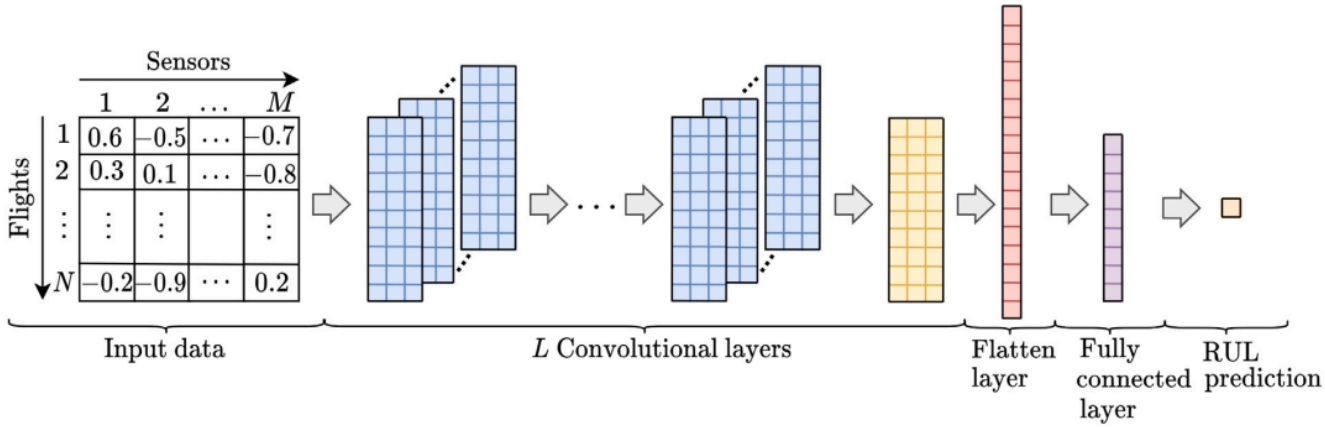


Fig. 14: Schematic overview of the RUL model [28].

TABLE II: Overview of RUL Model Experiments

Experiment	RUL Training Data Type	RUL Training Set Size	GAN Training Set Size	Augmentation Amount
1. Data Replacement	Synthesized Real	Full *(249 Units)* Reduced *(1-248 Units)*	Full *(249 Units)* Reduced *(50 Units)*	N/A
	Fully Synthetic	1k, 2k, 5k, 10k, 20k	Full *(249 Units)* Reduced *(50 Units)*	N/A
2. Data Augmentation	Real FD004 + Fully Synthetic	Reduced (1-248 Units) + Augmented	Full *(249 Units)* Reduced *(50 Units)*	1k, 2k, 5k, 10k, 20k
3. Baseline vs. Augmented	Real FD004	Full *(249 Units)* Reduced *(1-248 Units)*	N/A	N/A
	Real FD004 + Fully Synthetic	Full *(249 Units)* Reduced *(1-248 Units)*	Full *(249 Units)* Reduced *(50 Units)*	1k

size down to 1 unit. It assesses the maximum achievable performance of data augmentation when looking at the setups of the earlier experiments. This final result should provide a proof of concept of whether or not the data augmentation experiment leads to an improvement in prognostic performance.

5.3 *Prognostic Evaluation Results*

In this section, we analyze the prognostic performance of the GAN's synthetic data. The RUL model is trained with the training set and evaluated on the test set for all experiments. The augmentation in this section is done with fully synthetic data, i.e. OCs are sampled from the training data and combined with RUL labels till failure for all experiments except those explicitly mentioned. All the experiment results in this section depict the average result of 100 runs to reduce uncertainty. Furthermore, the confidence intervals (CIs) are given to allow fair comparison. This section is structured as follows. First, Section 5.3.1 evaluates the quality of synthetic data by analyzing the performance of RUL models trained exclusively on synthetic datasets, replacing real data entirely. Next, Section 5.3.2 explores the effect of augmenting real datasets with synthetic data across various configurations to identify the most effective augmentation strategy. This involves testing multiple sets of synthetic data with varying sizes and inherently different OCs. Using the optimal augmentation configuration identified, Section 5.3.3 compares the data augmentation setup against a baseline RUL model trained exclusively on real data, demonstrating the performance enhancements achieved through synthetic data augmentation.

5.3.1 *Synthetic Data Replacement*

We replace the RUL model's real training data with two types of synthetic data: synthesized real data and fully synthetic data. In Table III, the results of the data replacement experiments are depicted, where the RUL model is solely trained on synthetic data and thus no real data.

TABLE III: RUL model RMSE results evaluated on the test set for different GAN configurations where the training data is fully replaced with varying amounts and types of synthetic data.

Config.	Training Data	Augment Amount	RMSE
50-unit GAN	Synthesized Real	249 Units	**43.01**
	Fully Synthetic	1k	47.31
		2k	44.64
		3k	44.81
		4k	45.03
		5k	45.31
		7k	46.32
		10k	45.80
249-unit GAN	Synthesized Real	249 Units	28.08
	Fully Synthetic	1k	37.06
		2k	**26.82**
		3k	27.28
		4k	27.67
		5k	27.77
		7k	27.76
		10k	28.22

The results in Table III highlight the importance of GAN training diversity for synthetic data replacement. For both GANs, the potential of fully synthetic data is maximized at around 2k synthetic sequences augmentation, after which performance degrades, indicating overfitting to the GAN's synthetic training data.

The RUL model yields RMSE values of approximately 45 when it is trained exclusively with fully synthetic data generated by the 50-unit GAN. Synthesized real data slightly outperforms the fully synthetic data, likely because the 50-unit GAN relies on memorized RUL-OC combinations from its limited training set. This shows that the 50-unit GAN struggles to capture diverse degradation patterns.
In contrast, the 249-unit GAN demonstrates significantly better performance, achieving RMSE as low as 26.82 with $2k$ sequences augmentation. Unlike the 50-unit GAN, fully synthetic data outperforms the synthesized real data. This indicates that the 249-unit GAN generalizes better across unseen OC-RUL combinations. However, the underperformance of synthesized real data suggests overfitting certain dominant health degradation patterns within the real data. Overall, the 249-unit GAN outperforms the 50-unit GAN, highlighting the importance of a diverse training set for the GAN to generate diverse data for data replacement.

5.3.2 *Synthetic Data Augmentation*

In this section, we augment the training set of a RUL model with different configurations of fully synthetic to find the maximized benefit of data augmentation with the proposed GAN.

The results, shown in Table IV, depict the performance of RUL models augmented with synthetic data generated by the GAN trained on 50 units and 249 units. The RUL models were augmented with 10.000 varying fully synthetic sequences. The boundaries shown in the table represent the 95% CIs, and the bolded values represent results where the CIs do not overlap and are thus statistically significant.

TABLE IV: RMSE and 95% CIs for RUL Models Augmented with 10.000 fully synthetic sequences generated by GANs trained on 50 and 249 units across different training sizes.

Training Size (Real)	50 Units GAN aug	249 Units GAN aug
0 Units	47.37 ± 2.00	**28.09 ± 0.64**
1 Unit	35.40 ± 2.79	**27.67 ± 1.06**
3 Units	31.42 ± 2.55	**26.10 ± 1.20**
7 Units	29.19 ± 1.82	**25.14 ± 0.89**
12 Units	26.50 ± 1.12	**24.31 ± 0.68**
25 Units	23.71 ± 0.68	23.47 ± 0.52
50 Units	21.24 ± 0.52	21.63 ± 0.41
75 Units	19.98 ± 0.39	20.55 ± 0.38
100 Units	19.31 ± 0.33	19.23 ± 0.34
125 Units	18.89 ± 0.29	18.96 ± 0.27
150 Units	18.61 ± 0.24	18.96 ± 0.27
175 Units	18.46 ± 0.21	18.62 ± 0.28
200 Units	18.29 ± 0.24	18.45 ± 0.22
225 Units	18.27 ± 0.18	18.28 ± 0.21
249 Units	18.08 ± 0.17	18.23 ± 0.19

The table reveals that the model trained on 249 units demonstrates more stable performance with limited training data. This is particularly noticeable with very low amounts of real training data, where the 249-unit GAN outperforms the 50-unit GAN, especially in the case of full data replacement. This performance gap highlights the robustness of the 249-unit GAN in generating high-quality synthetic data that closely aligns with the real data. However, as the training size increases, the performance gap between the 50-unit GAN and the 249-unit GAN diminishes. Despite overlapping CIs, the 50-unit GAN consistently outperforms the 249-unit GAN for real training sizes over 50 units. This may be because the synthetic data generated by the 50-unit GAN contains noticeable deficiencies that the RUL model can learn to disregard. In doing so, the RUL model may still pick up on subtle patterns that improve the RUL model's prognostic performance. In contrast, the minor inaccuracies of the 249-unit GAN's realistic data may be harder for the RUL model to pick up on and disregard. Consequently, the RUL model learns from these minor inaccuracies, slightly decreasing its prognostic performance. As the size of real training data increases and the RUL model becomes stronger, it becomes less focused on the synthetic data, diminishing the difference between the two configurations.

From a practical perspective, using a GAN trained on 249 units to augment a dataset with fewer than 25 units is unrealistic in real-world scenarios. Therefore, from this point forward, we use the GAN trained on 50 units for data augmentation experiments to ensure meaningful experiments unless explicitly mentioned.

To investigate how the amount of synthetic sequence augmentation influences the RUL model performance, we experimented with varying amounts of data augmentation. In, Figure 15, the results of a RUL model augmented with 1,000 and 20,000 varying synthetic sequences are depicted. The 1,000-sequence augmentation is included because it demonstrates strong performance in the experiments, while the 20,000-sequence augmentation provides a clear contrast, showcasing the impact of a significantly larger augmentation amount.

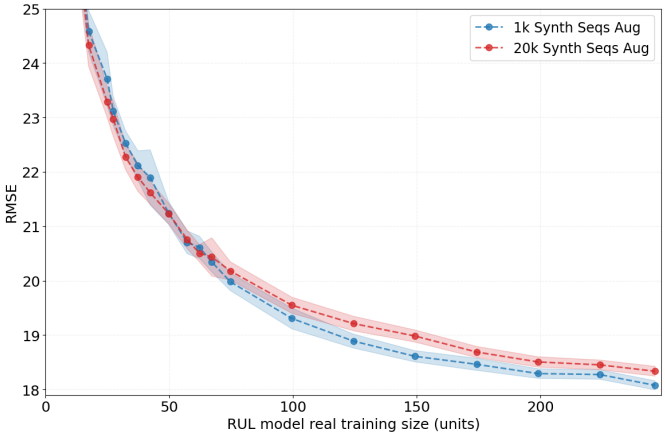


Fig. 15: Comparison of RMSE for RUL models trained on varying real training sizes, augmented with 1,000 and 20,000 synthetic sequences. The shaded regions represent the 95% CIs

As illustrated in Figure 15, the amount of synthetic augmentation amount influences the prognostic performance of the RUL model. For smaller training sizes, high amounts of data augmentation results in more performance enhancement. However as the RUL model's training set size grows, and its prognostic performance without augmentation improves, the amount beneficial for augmentation for prognostic performance enhancement diminishes. At smaller real dataset sizes, heavier augmentation leads to larger performance gains. However, as the real training set grows, lighter augmentation becomes more effective.

In addition to the number of sequences used for augmentation, the specific sequences chosen to augment also greatly influence the RUL predictive performance. As shown in Figure 16, the choice of synthetic sequences significantly impacts model outcomes. Both RUL models were trained on the same real training set and augmented with 1,000 fully synthetic sequences. However, the 1,000 synthetic sequences for augmentation differed between the two models. While sets of sequences share a similar number of units until failure, the combinations of RUL-OC sequences differ for the two sets.

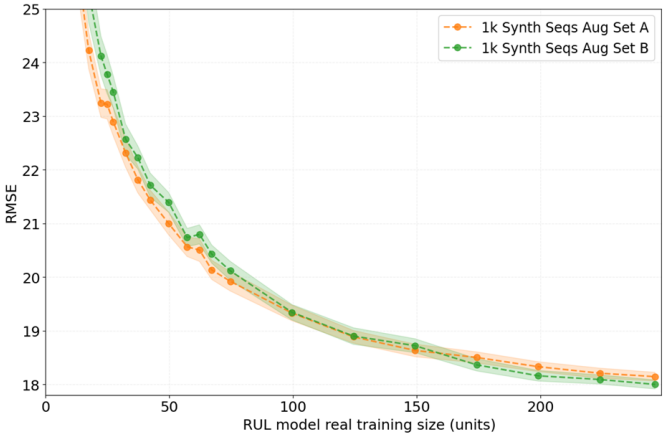


Fig. 16: Two cases of RMSE of RUL models for varying training sizes, augmented with two different sets of 1000 synthetic sequences generated by the 50-unit GAN.

The results demonstrate variability across the different sets of synthetic sequences. Set A outperforms set B for smaller real training sizes (fewer than 100 units), indicating that set A provides better quality augmentation when the RUL model has limited real data and relies more heavily on synthetic sequences. Conversely, as the real training set size increases, set B begins to outperform set A, suggesting that the RUL model leverages certain patterns in set B that enhance performance as it becomes stronger. This shift in performance with increasing training size implies that set A is more beneficial when the RUL model is less robust, providing utility in data-scarce scenarios. In contrast, set B demonstrates its strengths as the RUL model becomes more advanced and selective about the data it learns from.

5.3.3 *Baseline vs. Augmented*

In this section, we evaluate the impact of data augmentation on the RUL model by comparing augmentation configurations of the RUL model using the 50-unit GAN against a baseline model trained solely on real data. This analysis determines whether data augmentation of synthetic data by our GAN provides meaningful improvements in predictive performance. The baseline RUL model, trained without synthetic data, serves as a benchmark for assessing the effectiveness of the augmentation. The selected configurations represent amounts and sets of synthetic sequences that performed well in the experiments of Section 5.3.2. Additionally, we investigate the impact of data augmentation on the RUL model's specific predictive performance, identifying RUL ranges where the augmented model outperforms or underperforms compared to the baseline. This answers whether we improve the predictive performance of the baseline RUL model when its training set is augmented with synthetic data generated by our proposed GAN.

The results in Table V compare the RMSE values of models augmented with 1,000 synthetic sequences from two distinct sets: Set A and Set B as discussed in the previous section. The table includes the RMSE's 95% CIs and the relative percentage change in RMSE for each augmented model compared to the baseline. Statistically significant improvements, where the CI of the augmented case has minimal or no overlap with the baseline CI, are highlighted in bold. Both augmentation sets contain a similar number of units with comparable mean lifetimes, and all RUL models are trained on identical real datasets. Furthermore, the GANs used for augmentation are trained on the same 50 units. Notably, we are solely showcasing the results of the 50-unit GAN augmentation, the 249-unit GAN augmentation results were comparable and are therefore omitted.

As evident from Table V, Set A consistently improves the baseline RUL on average up to 150 real training units, with the largest improvements observed for smaller training sizes. As shown in Figure 17, even in cases where the 95% CIs of the augmented model and the baseline overlap, the RMSE values for Set A remain consistently lower than those of the baseline. In regions with even higher data scarcity, up until 31 units, the increased prognostic performance of the RUL model augmented with Set A synthetic data is statically significant as the CIs do not overlap. This indicates that set A provides meaningful data augmentation in data-scarce regions where the RUL model is less robust. Importantly, this augmentation is achieved with a GAN trained on only 50 units, reinforcing the feasibility of a data-sharing framework where multiple entities contribute small datasets to collaboratively train a general GAN, achieving meaningful data augmentation improvements without direct data exchange.

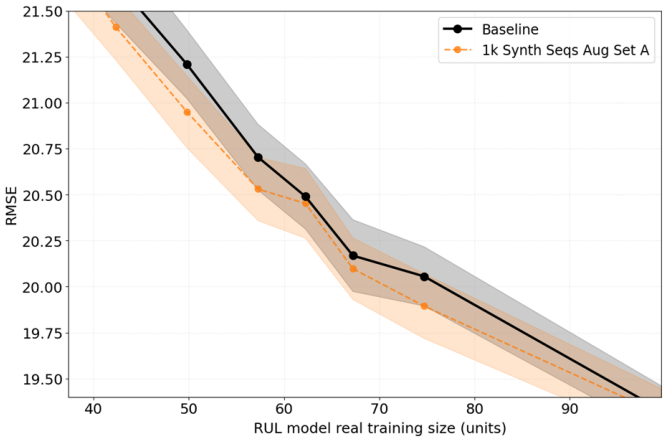


Fig. 17: RMSE of a RUL model with varying training sizes with 1000 synthetic sequences augmentation from set A vs without (Baseline).

Set B on the other hand exhibits a distinct performance pattern. As evident from the 95% CIs in Figure 18, the RMSE gains become statistically significant at larger training sizes (above 175 units). Unlike Set A, which primarily benefits smaller datasets, Set B is more effective when the RUL model has access to more real training data. This suggests that Set B captures a more complex degradation pattern that the RUL model can only leverage as it becomes more robust with sufficient real training data. This statistically significant improvement of approximately 1% is particularly notable given that the GAN is only trained on 50 units. Therefore, the observed prognostic improvement is practical and relevant for real-world scenarios, showcasing the Super-Space Time GAN's utility, even when trained in a data-scarce setting. The performance enhancement represents meaningful improvements for a state-of-the-art RUL model. With further refinements in GAN training and more careful synthetic sequence selection, the performance gains could be even more substantial.

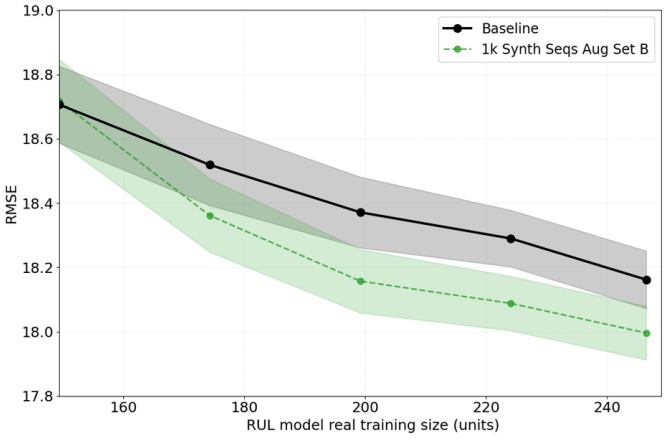


Fig. 18: RMSE of a RUL model with varying training sizes with 1000 synthetic sequences augmentation from set B vs without (Baseline).

Overall, these results highlight the utility of the proposed Super-

TABLE V: RMSE, 95% CIs, and percentage change relative to the baseline for RUL models with different sets of 1,000 synthetic sequence augmentation generated by the 50-unit GAN.

Training Size (Real)	Baseline	1k Seq Aug Set A	1k Seq Aug Set B
0 Units	N/A	**46.85 ± 0.88**	49.51 ± 1.30
1 Unit	55.09 ± 1.66	**31.91 ± 0.93 (-42.07%)**	39.47 ± 1.59 (-28.34%)
7 Units	30.39 ± 0.58	**28.80 ± 0.56 (-5.24%)**	31.13 ± 0.60 (+2.43%)
12 Units	27.09 ± 0.48	**26.00 ± 0.46 (-4.03%)**	27.36 ± 0.49 (+1.00%)
17 Units	25.09 ± 0.40	**24.22 ± 0.39 (-3.47%)**	25.31 ± 0.41 (+0.88%)
22 Units	24.12 ± 0.35	**23.24 ± 0.34 (-3.64%)**	24.20 ± 0.36 (+0.33%)
25 Units	23.87 ± 0.34	**23.22 ± 0.33 (-2.72%)**	23.78 ± 0.35 (-0.04%)
31 Units	22.91 ± 0.30	**22.29 ± 0.29 (-2.71%)**	22.57 ± 0.31 (-1.49%)
37 Units	22.00 ± 0.28	21.81 ± 0.27 (-0.86%)	22.22 ± 0.29 (+1.00%)
42 Units	21.71 ± 0.26	21.43 ± 0.25 (-1.29%)	21.71 ± 0.27 (0.00%)
50 Units	21.25 ± 0.25	21.00 ± 0.24 (-1.18%)	21.39 ± 0.26 (+0.66%)
57 Units	20.56 ± 0.22	20.50 ± 0.21 (-0.29%)	20.73 ± 0.24 (+0.83%)
63 Units	20.52 ± 0.21	20.50 ± 0.20 (-0.10%)	20.79 ± 0.23 (+1.32%)
69 Units	20.42 ± 0.19	20.13 ± 0.18 (-1.42%)	20.42 ± 0.19 (0.00%)
75 Units	20.08 ± 0.16	19.92 ± 0.17 (-0.80%)	20.11 ± 0.18 (+0.15%)
100 Units	19.34 ± 0.16	19.33 ± 0.12 (-0.05%)	19.34 ± 0.13 (0.00%)
125 Units	18.96 ± 0.11	18.88 ± 0.12 (-0.42%)	18.90 ± 0.13 (-0.32%)
150 Units	18.71 ± 0.12	18.63 ± 0.12 (-0.43%)	18.72 ± 0.13 (+0.05%)
175 Units	18.52 ± 0.12	18.50 ± 0.10 (-0.11%)	**18.36 ± 0.11 (-0.86%)**
200 Units	18.37 ± 0.11	18.33 ± 0.09 (-0.22%)	**18.16 ± 0.09 (-1.14%)**
225 Units	18.29 ± 0.09	18.21 ± 0.09 (-0.44%)	**18.09 ± 0.08 (-1.09%)**
249 Units	18.16 ± 0.09	18.14 ± 0.08 (-0.11%)	**17.99 ± 0.08 (-0.93%)**

SpaceTime GAN for synthetic augmentation. Set A proves effective across various training sizes, particularly in data-scarce conditions. Set B meanwhile excels at larger training sizes, showing statistically significant improvements beyond 175 units, which is larger than the 50 units the GAN is trained on. This highlights the practical significance of these improvements, as the synthetic sequences successfully enhance the RUL model's prognostic performance, even in areas where the real data availability exceeds the GAN's training set. Furthermore, the performance gap between Set A and Set B underscores the importance of tailoring synthetic augmentation strategies to the available real data and the robustness of the baseline model.

Error per RUL

We compare the baseline RMSE per RUL with the RMSE per RUL of the augmented case to identify areas where the synthetic sequences enhance performance and where they may lead to a decline. Here, RMSE per RUL refers to the error calculated specifically for predictions corresponding to each RUL value, which are grouped in RUL brackets with a span of 10. In Figure 19, the baseline model is trained on the full dataset, and the deltas shown represent the same training set augmented with 20,000 synthetic sequences generated by the GAN trained on 50 units. Similarly, Figure 20 illustrates the results for the same setup but using synthetic sequences generated by the 249-unit GAN augmentation. Note that for both figures, the training size in combination with 20,000 sequence augmentation does not lead to an increase in prognostic performance. We employed a rather large augmentation size of 20,000 to amplify the difference between a strong baseline and augmented models.

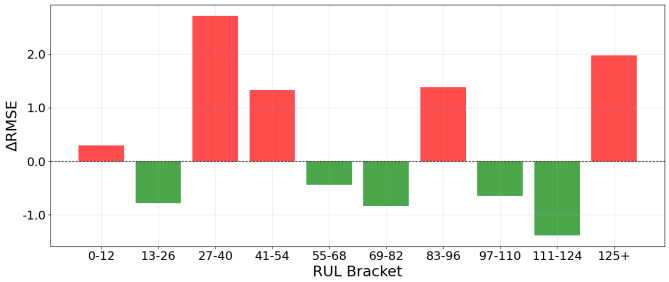


Fig. 19: Change in RMSE of baseline RUL model trained on the full training set compared to augmented case (augmented with 20,000 sequences from the 50-unit GAN).

The RUL model augmented with sequences from the 50-unit GAN consistently increases the RMSE for units in the lower RUL brackets RUL < 54, indicating weaker performance for predictions near failure. However, the augmentation often decreases the RMSE for RUL medium to medium-high RUL brackets ($RUL \in [55, 124]$), suggesting an increase in performance for predicting early to mid-degraded phases.

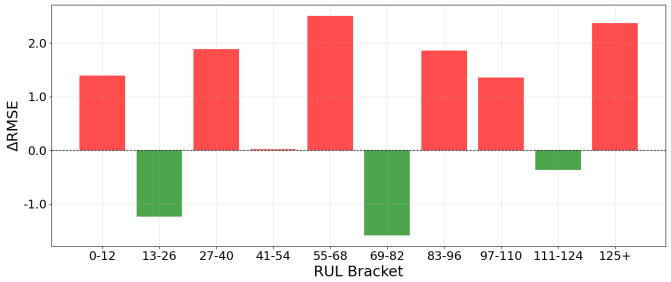


Fig. 20: Change in RMSE of baseline RUL model trained on the full training set compared to augmented case (augmented with 20,000 sequences from the 249-unit GAN).

Augmenting the RUL model's training set with sequences generated by the 249-unit GAN generally increases RMSE across most RUL brackets, suggesting a decline in overall performance. However, improvements are observed in the 13-26 and 69-82 RUL brackets, indicating that the generated sequences enhance prognostic accuracy in those specific regions. This pattern suggests that the GAN may have overfitted to certain degradation trends, leading to localized benefits while reducing generalizability across the full degradation spectrum. This is consistent with the findings about the synthetic data quality in Section 4.2, which stated that the GAN overfitted to certain health degradation patterns.

5.3.4 Challenges & Limitations

In this section, we address the main challenges and limitations encountered during the prognostic evaluation of the proposed Super-SpaceTime GAN. Regarding the Super-SpaceTime GAN's data augmentation, we found two limitations:

1) **Prognostic performance-based GAN tuning:** The GAN is tuned primarily based on data quality metrics such as the Jenson-Shannon Distance. The absence of a direct consideration of the synthetic data's prognostic performance limited its practicality.

2) **Dynamic augmentation strategies:** The synthetic data showed large variability in data augmentation effectiveness. Certain OC-RUL combinations enhanced the RUL model's prognostic performance while some diminished it. Furthermore, the effectivity of synthetic sequences was dependent on the training size.

Addressing these issues is crucial to maximizing the impact of the synthetic data on RUL model performance. Incorporating prognostic performance-based tuning and more selective augmentation strategies will ensure that generated sequences consistently enhance model accuracy. Overall, this will further increase the prognostic performance of the RUL models augmented with the Super-SpaceTime GAN's synthetic data.

6 Conclusion

In this study, we have proposed a **Super-SpaceTime GAN**, a supervised GAN conditioned on operating conditions (OCs) and Remaining Useful Life (RUL) labels, to generate synthetic condition monitoring (CM) data. Key modifications to the GAN include:

- Incorporating an autoencoder to learn efficient and more guided latent space representation, preserving key temporal and spatial dependencies while reducing noise.
- Conditioning on OC and RUL labels to produce data relevant to specific operational and health states.
- Adding a supervised loss, which is altered to align the synthetic data more closely with the temporal and spatial dependencies of RUL prediction tasks.

The proposed Super-SpaceTime GAN was thoroughly evaluated for its synthetic data quality. The evaluation consisted of a visual inspection by PCA and t-SNE and a novel metric that employs an autoencoder on the synthetic data to evaluate the health degradation. We found that the synthetic data aligns well with the real data but showed certain weaknesses such as mode collapse and exaggerated degradation patterns, highlighting areas for further refinement.

Furthermore, we trained a state-of-the-art RUL model augmented with the synthetic data to evaluate the prognostic performance of the synthetic data. We found a practical, real-world, applicable enhancement in prognostic performance using a GAN trained on a small dataset of 50 units. Here, we observed a 0.9% improvement in the performance of the RUL model trained on the full dataset and a 1.2% increase in prognostic performance for the RUL model trained on a limited dataset of 50 units. Additionally, we found great potential for data augmentation at even smaller training set sizes, as the augmentation benefit increased as the training size diminished. This reinforces the viability of leveraging a shared GAN trained on pooled datasets, allowing individual airliners to enhance their RUL models without direct data exchange.

By addressing identified challenges and leveraging the potential of synthetic data, this study provides strong evidence for using data augmentation to boost prognostic performance in RUL prediction models. The results underline the importance of tailoring GAN configurations and augmentation strategies to maximize their utility in diverse operational scenarios.

7 Future Work

Conditioning the GAN on Health Indices (HI) could address current limitations by promoting diverse degradation trajectories and better generalization to unseen scenarios. This approach may enhance the generation of realistic and robust synthetic data for RUL model training. Two potential configurations are proposed:

Supervised HI Conditioning: Incorporating HI alongside RUL and OCs leverages the relationship between HI and RUL, ensuring consistency in degradation patterns. This approach is ideal for further increasing the prognostic performance if ample labeled data is available but may face challenges with sparse datasets.

Unsupervised HI Conditioning: Replacing RUL labels with HI allows the GAN to generate synthetic data without requiring RUL labels. The HI can be estimated using autoencoders or other models, enabling the GAN to generalize degradation patterns, especially in unsupervised settings.

Future work should also focus on refining tuning and augmentation processes, such as dynamic augmentation strategies tailored to real dataset size and composition. Integrating prognostic performance metrics, such as RUL prediction accuracy, into GAN tuning could align synthetic data generation with application-specific goals, maximizing its impact on RUL model performance. Additionally, exploring the training of GAN on even smaller datasets, with fewer than 50 units, is a valuable direction for future work, as data augmentation becomes more beneficial as the size of the real dataset decreases.

Acknowledgements

I want to thank my esteemed supervisors, Ingeborg de Pater and Kristupas Bajarunas, for their splendid guidance and insightful comments which greatly improved my research endeavors and paper. Furthermore, I want to thank my beloved twin brother Luc for lending me his laptop in times of computational despair.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[2] A. Radford, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[3] M. Zhan, J. Fan, and J. Guo, "Generative adversarial inverse reinforcement learning with deep deterministic policy gradient," *IEEE Access*, 2023.

[4] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *Advances in neural information processing systems*, vol. 29, 2016.

[5] E. Brophy, Z. Wang, Q. She, and T. Ward, "Generative adversarial networks in time series: A systematic literature review," *ACM Computing Surveys*, vol. 55, no. 10, pp. 1–31, 2023.

[6] O. Mogren, "C-RNN-GAN: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016.

[7] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," *Advances in neural information processing systems*, vol. 32, 2019.

[8] S. Chattoraj, S. Pratiher, S. Pratiher, and H. Konik, "Improving stability of adversarial Li-ion cell usage data generation using generative latent space modelling," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 8047–8051.

[9] C.-Y. Tai, W.-J. Wang, and Y.-M. Huang, "Using time-series generative adversarial networks to synthesize sensing data for pest incidence forecasting on sustainable agriculture," *Sustainability*, vol. 15, no. 10, p. 7834, 2023.

[10] H. Shi, Y. Xu, B. Ding, J. Zhou, and P. Zhang, "Long-term solar power time-series data generation method based on generative adversarial networks and sunrise–sunset time correction," *Sustainability*, vol. 15, no. 20, p. 14920, 2023.

[11] L. Mushunje, D. Allen, and S. Peiris, "Volatility and irregularity capturing in stock price indices using time series generative adversarial networks (TimeGAN)," *arXiv preprint arXiv:2311.12987*, 2023.

[12] I. Annaki, M. Rahmoune, and M. Bourhaleb, "Overview of data augmentation techniques in time series analysis." *International Journal of Advanced Computer Science & Applications*, vol. 15, no. 1, 2024.

[13] G. Iglesias, E. Talavera, Á. González-Prieto, A. Mozo, and S. Gómez-Canaval, "Data augmentation techniques in time series domain: a survey and taxonomy," *Neural Computing and Applications*, vol. 35, no. 14, pp. 10 123–10 145, 2023.

[14] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are GANs created equal? a large-scale study," *Advances in neural information processing systems*, vol. 31, 2018.

[15] T. Pan, J. Chen, T. Zhang, S. Liu, S. He, and H. Lv, "Generative adversarial network in mechanical fault diagnosis under small sample: A systematic review on applications and future perspectives," *ISA transactions*, vol. 128, pp. 1–10, 2022.

[16] P. Lang, K. Peng, J. Cui, J. Yang, and Y. Guo, "Data augmentation for fault prediction of aircraft engine with generative adversarial networks," in *2021 CAA Symposium on Fault Detection, Supervision, and Safety for Technical Processes (SAFEPROCESS)*. IEEE, 2021, pp. 1–5.

[17] X. Zhang, Y. Qin, C. Yuen, L. Jayasinghe, and X. Liu, "Time-series regeneration with convolutional recurrent generative adversarial network for remaining useful life estimation," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 6820–6831, 2020.

[18] R. He, Z. Tian, and M. J. Zuo, "A semi-supervised GAN method for RUL prediction using failure and suspension histories," *Mechanical Systems and Signal Processing*, vol. 168, p. 108657, 2022.
[19] K. Rombach, G. Michau, and O. Fink, "Controlled generation of unseen faults for partial and open-partial domain adaptation," *Reliability Engineering & System Safety*, vol. 230, p. 108857, 2023.
[20] F. Zhou, S. Yang, H. Fujita, D. Chen, and C. Wen, "Deep learning fault diagnosis method based on global optimization GAN for unbalanced data," *Knowledge-Based Systems*, vol. 187, p. 104837, 2020.
[21] K. Yan, J. Su, J. Huang, and Y. Mo, "Chiller fault diagnosis based on VAE-enabled generative adversarial networks," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 387–395, 2020.
[22] J. Xiong, O. Fink, J. Zhou, and Y. Ma, "Controlled physics-informed data generation for deep learning-based remaining useful life prediction under unseen operation conditions," *Mechanical Systems and Signal Processing*, vol. 197, p. 110359, 2023.
[23] K. Bajarunas, M. L. Baptista, K. Goebel, and M. A. Chao, "Health index estimation through integration of general knowledge with unsupervised learning," *Reliability Engineering & System Safety*, vol. 251, p. 110352, 2024.
[24] A. Kiran and S. S. Kumar, "A methodology and an empirical analysis to determine the most suitable synthetic data generator," *IEEE Access*, 2024.
[25] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.
[26] A. Saxena and K. Goebel, "Turbofan engine degradation simulation data set," *NASA ames prognostics data repository*, vol. 18, pp. 878–887, 2008.
[27] H. Li, W. Zhao, Y. Zhang, and E. Zio, "Remaining useful life prediction using multi-scale deep convolutional neural network," *Applied Soft Computing*, vol. 89, p. 106113, 2020.
[28] I. de Pater, A. Reijns, and M. Mitici, "Alarm-based predictive maintenance scheduling for aircraft engines with imperfect remaining useful life prognostics," *Reliability Engineering & System Safety*, vol. 221, p. 108341, 2022.
[29] M. Ma and Z. Mao, "Deep-convolution-based LSTM network for remaining useful life prediction," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 1658–1667, 2020.
[30] J. Xia, Y. Feng, C. Lu, C. Fei, and X. Xue, "LSTM-based multi-layer self-attention method for remaining useful life estimation of mechanical systems," *Engineering Failure Analysis*, vol. 125, p. 105385, 2021.
[31] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. 11, 2008.
[32] P. Malhotra, V. Tv, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Multi-sensor prognostics using an unsupervised health index based on lstm encoder-decoder," *arXiv preprint arXiv:1608.06154*, 2016.
[33] F. Xu, Z. Huang, F. Yang, D. Wang, and K. L. Tsui, "Constructing a health indicator for roller bearings by using a stacked auto-encoder with an exponential function to eliminate concussion," *Applied Soft Computing*, vol. 89, p. 106119, 2020.
[34] D. She, M. Jia, and M. G. Pecht, "Sparse auto-encoder with regularization method for health indicator construction and remaining useful life prediction of rolling bearing," *Measurement Science and Technology*, vol. 31, no. 10, p. 105005, 2020.
[35] S. Zhai, B. Gehring, and G. Reinhart, "Enabling predictive maintenance integrated production scheduling by operation-specific health prognostics with generative deep learning," *Journal of Manufacturing Systems*, vol. 61, pp. 830–855, 2021.
[36] H. Lee, H. J. Lim, and A. Chattopadhyay, "Data-driven system health monitoring technique using autoencoder for the safety management of commercial aircraft," *Neural Computing and Applications*, vol. 33, no. 8, pp. 3235–3250, 2021.
[37] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *2008 international conference on prognostics and health management*. IEEE, 2008, pp. 1–9.