Advanced Engineering Informatics 74 (2026) 104685

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Full length article

Engine-specific degradation prediction of aviation engines via transferable snippet augmentation: A trend-grouped fine-tuning perspective

Haoze Wu [a], Shisheng Zhong [a,b,c,*], Minghang Zhao [b,c], Yongjian Zhang [b,c,*], Xuyun Fu [b,c], Song Fu [a]

[a] School of Mechatronics Engineering, Harbin Institute of Technology, Harbin 150001, China
[b] Department of Mechanical Engineering, Harbin Institute of Technology, Weihai 264209, China
[c] Weihai Key Laboratory of Intelligent Operation and Maintenance, Harbin Institute of Technology, Weihai 264209, China

ARTICLE INFO

Keywords:
Aviation engine gas path performance
Temporal forecasting
Transferable snippet augmentation
Exhaust gas temperature

ABSTRACT

Accurate prediction of key gas path performance parameters plays a crucial role in informing subsequent maintenance planning for aero-engine operations. A basic model trained on historical data from all engines often fails to capture engine-specific characteristics, leading to limited prediction accuracy. Mainstream approaches address this by fine-tuning the basic model for each individual engine. However, this per-engine adaptation greatly increases maintenance complexity and performs poorly when only limited data are available for a new engine. To address this challenge, this paper proposes the Transferable Snippet Augmentation Network (TSAN), which shifts model adaptation from engine-level fine-tuning to degradation-trend-level specialization. The key idea is to use transferable degradation snippets. They are short segments from different engines that share similar degradation tendencies and help improve prediction accuracy when engine-specific data are limited. TSAN operates in two stages. First, a unified base model is trained on all engines. Second, the snippets are embedded and clustered into degradation-trend groups, which enables group-wise fine-tuning and produces specialized predictors. At inference, the degradation-trend group of each sample is identified to select the appropriate specialized model. Experimental results show that the proposed TSAN reduces the number of required models by 53% and achieves improvements of 36.0%, 35.7%, and 34.1% over baseline methods in Mean Absolute Error (MAE), Mean Relative Error (MRE), and Root Mean Square Error (RMSE), respectively. It also delivers 18.0%, 11.1%, and 15.2% gains in prediction stability. Ablation studies further validate the effectiveness of the network design, the degradation-trend grouping strategy, and the feasibility of the proposed fine-tuning perspective.

1. Introduction

Driven by the concept of prognostics and health management, the maintenance strategy of aviation engines has gradually shifted from regular maintenance to predictive maintenance. During multiple flight cycles of a single aircraft, the performance of an individual engine gradually declines as the number of flight cycles increases. To meet the thrust requirements during the take-off phase, a deteriorating engine typically exhibits higher exhaust gas temperatures (EGT) [1]. The increase in EGT accelerates the degradation of gas path components such as high-pressure compressors and turbine blades, thereby raising their failure rates [2]. Consequently, in recent years, many studies have focused on predicting the key gas path performance parameters, with the aim of mitigating potential risks arising from adverse environmental and operational conditions, including the works by Jung [3], Turgut [4], and Xiao [5].

To predict the gas path performance parameters of an aviation engine, an ideal approach is to model individual components separately and then combine them into an overall performance model. This model accounts for interactions among components under specific operating conditions. This approach, known as the component diagram method [6], can achieve high prediction accuracy [7]. However, building such models remains difficult. The mapping relationships among components are not fully established, making the modeling process time-consuming and costly [8]. Aviation engines are complex and highly nonlinear systems. To improve performance, safety, and efficiency, their structures have become more intricate. As a result, mechanism-based methods

* Corresponding authors at: Department of Mechanical Engineering, Harbin Institute of Technology, Weihai 264209, China.
E-mail addresses: 21b908077@stu.hit.edu.cn (H. Wu), zhongss@hit.edu.cn (S. Zhong), zhaomh@hit.edu.cn (M. Zhao), zhangyj@hitwh.edu.cn (Y. Zhang), fuxuyun@hit.edu.cn (X. Fu), fusong@hit.edu.cn (S. Fu).

https://doi.org/10.1016/j.aei.2026.104685
Received 4 December 2025; Received in revised form 27 February 2026; Accepted 7 April 2026
Available online 15 April 2026

Nomenclature

TSAN	Transferable Snippet Augmentation Network
MAE	Mean Absolute Error
MRE	Mean Relative Error
RMSE	Root Mean Square Error
EGT	Exhaust Gas Temperature
RNN	Recurrent Neural Network
EGTM	Exhaust Gas Temperature Margin
GRU	Gated Recurrent Unit
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
OPTICS	Ordering Points to Identify the Clustering Structure
GMM	Gaussian Mixture Model
MSE	Mean Squared Error
BIC	Bayesian Information Criterion
OEM	Original Equipment Manufacturer
ACARS	Aircraft Communications Addressing and Reporting System
DTW	Dynamic Time Warping
EGTM_S	Smoothed EGTM
MTGNN	Multivariate Time Series Forecasting with Graph Neural Networks
TSAN-NC	TSAN without the clustering procedure
y	Actual values of performance parameters
θ	Intrinsic engine state
L	Sampling length
m	Index of the flight whose data serve as the prediction target
$\boldsymbol{x}_m$	Key parameters of the target flight
n	Index of the flight to be predicted
$\boldsymbol{h}_1$	Hidden state output from the GRU in the representation module
$\boldsymbol{h}_2$	Hidden state output from the Transformer layer in the representation module
$\widetilde{y}$	Predicted value from the prediction module
α	Weight balancing prediction and optimization in the trend representation module
$\boldsymbol{S}_m$	Latent degradation state of the engine at flight cycle m
ε	Estimation noise between two parts in representation module
$\mathcal{N}(0,\sigma^2 I)$	Multivariate Gaussian distribution with variance $\sigma^2 I$
$\mathscr{L}_1$	Loss of the representation module
N	Total number of engines
$\boldsymbol{X_c}$	Inputs to the representation module for generating representation
T	Total number of training samples
$\boldsymbol{l}$	Vector of cluster labels for the T samples
$\boldsymbol{H}_c$	Trend representations output for each sample
K	Number of clusters
ρ	BIC improvement ratio
τ	Threshold for the improvement ratio
$\mathscr{L}_2$	Loss of the prediction module
$\boldsymbol{X}_l, \boldsymbol{X}_r$	Left and right segments of the TSAN input
$\boldsymbol{Z}$	Transformed features obtained by passing X_r through two Transformer encoder layers
$\boldsymbol{H}$	Hidden states obtained from the Transformer encoder and a linear layer (corresponding to the hidden state in Fig. 3)
$\boldsymbol{w}$	Weight vectors obtained from the Transformer encoder and a linear layer (corresponding to the weight in Fig. 3)
$\boldsymbol{z}$	Zero vector
$\boldsymbol{X}_{tmp}$	Concatenation of $\boldsymbol{X}_r$ and $\boldsymbol{x}_m$
$\boldsymbol{h}_m$	Hidden representation at timestamp m produced by the GRU
T_{SLOAT}	Sea Level Outside Air Temperature
ALT	Altitude
T_{TIAT}	Total Inlet Air Temperature
M	Mach Number
N_1	Fan Speed
N_2	Core Engine Speed
C	Number of flight cycles
X_n	Data corresponding to the n-th engine
$\widetilde{\boldsymbol{y}}$	Final prediction vector
$\boldsymbol{y}$	Ground truth vector
i	Index of an element in a vector
y_i	i-th element of $\boldsymbol{y}$
$\widetilde{\boldsymbol{y}}^{avg}$	Average of 10 prediction results
$\widetilde{\boldsymbol{y}}_s$	Values used for visualizing differences in plots

often face issues of scalability and modeling difficulty [9]. Even engines with the same design show variations due to dimensional tolerances and operating conditions. Component replacement during maintenance adds further uncertainty. These factors make mechanism-based modeling difficult and require deep domain expertise from engineers. Therefore, researchers increasingly turn to data-driven methods. Such approaches can capture nonlinear relationships directly from operational data and offer a practical alternative to traditional mechanism-based modeling.

The data-driven approach avoids explicit component-level modeling by leveraging the physical information inherent in operational data. It links environmental and control parameters with performance parameters that embed prior physical understanding to enable more reliable predictions [10]. Data-driven prediction methods can be generally divided into classical statistical methods and machine learning-based methods. To provide a brief overview of the research landscape, representative recent studies covering various modeling paradigms are summarized in Table 1. Classical approaches, such as Auto Regressive Integrated Moving Average [11], Kalman filtering [12], and support vector regression [13], are commonly employed in time series prediction. Moreover, Zaccaria et al. developed a digital twin-based framework for aviation engine monitoring and diagnostics, adopting an integrated physics-based and data-driven approach for engine performance management [14]. The framework also incorporates considerations for individual engine variability within the fleet. These methods have good interpretability, low data requirements, and few hyperparameters. However, they struggle to capture nonlinear dynamics and complex patterns, and their predictive performance has been surpassed by machine learning models [15,16]. In machine learning-based prediction, a neural network is constructed and trained on environmental and target parameters collected during engine operation. The loss between predicted and actual values is minimized through backpropagation, enabling the network to infer unknown gas path performance parameters. For example, Ilbas et al. [17] employed an artificial neural network to predict the EGT of the CFM56-7B engine. Dursun et al. [18] and Aygun et al. [19] applied LSTM networks to predict engine energy and emission parameters; Xiao et al. [20] employed a Recurrent Neural Network (RNN)-based model for EGT prediction. Yu et al. [21] proposed a *meta*-learning framework for cases with low-quality data. Nevertheless, existing machine learning approaches often ignore engine-specific characteristics and lifecycle variations, which constrains their prediction accuracy and generalization ability.

Recent studies have taken individual engine differences into account and explored personalized modeling approaches. Transfer learning offers an effective solution by adapting a pretrained base model to each specific engine through fine-tuning with limited task-related data [25]. The typical process involves: first, training a base model following the

Table 1
Summary table of recent papers on the prediction of aviation engine gas path deterioration.

Reference	Year	Modeling approach	Key characteristics	Performance
Lin et al. [22]	2021	Sample-adaptive LSTM	Handles limited data scenarios through adaptive framework at the engine level	Achieved superior short-term prediction stability in small-sample regimes
Yan et al. [23]	2022	Transfer learning	Explicitly models step responses to predict performance under maintenance-induced disturbances	Successfully captured sudden EGTM step-changes
Huang et al. [1]	2023	Stochastic dynamics	Estimates deterioration parameters robustly under multi-source uncertainties	High robustness in parameter observation against sensor noise and uncertainty
Xiao et al. [15]	2024	Physics-embedded LSTM	Integrates component-level physical topology for accurate degradation tracking	Accurate component-level degradation tracking
Jin et al. [9]	2024	Multi-resolution Transformer	Captures multi-scale temporal dependencies for component-level prediction	Accurate component-level forecasting across different time scales
He et al. [24]	2025	Transfer learning	Facilitates cross-engine estimation of performance degradation via engine-level modeling	Effective estimation of performance degradation in cross-engine tasks
Jung et al. [3]	2025	Maintenance-aware DNN	Incorporates maintenance segmentation for long-term degradation prediction	Improved prediction accuracy for long-term degradation
Xiao et al. [5]	2025	Spatio-temporal Digital Twin	Decouples spatial–temporal features while preserving physical structure integrity	Enhanced stability in performance prediction

standard neural network procedure; second, creating a copy of this model for each engine; third, fine-tuning it with the engine's historical data to obtain an engine-specific model; and finally, using these personalized models for prediction tasks [22]. This approach has been applied to various studies. For example, Yan et al. [23] used transfer learning to handle data scarcity. Liu et al. [26] applied it with simulated datasets for auxiliary power unit monitoring. He et al. [24] performed transfer learning across different engine operating conditions to improve the model's adaptability in available power estimation. Compared with using a unified base model, the fine-tuned models better capture engine-specific characteristics, thus achieving improved prediction accuracy. However, the transfer learning-based personalized modeling approach still faces several challenges. First, data imbalance leads to biased training, as engines with abundant data dominate model optimization while those with limited samples are prone to overfitting during fine-tuning [27]. Second, each engine requires an independent fine-tuned model, which significantly increases the complexity of model management and maintenance. Third, the method focuses on individual optimization and fails to exploit shared degradation information among engines, which limits its adaptability to newly deployed engines with scarce data and incomplete degradation patterns. Consequently, this approach lacks scalability and cannot effectively support large-scale and unified fleet management.

To address these three issues, this paper attempts to revisit the problem from the perspective of the prediction task itself, exploring whether the modeling process during transfer can be made more effective and whether the characteristics of engine performance and degradation can be shared across different engines at comparable stages. In aero-engine analysis, the performance parameters are fundamentally driven by the interplay of environmental conditions, operating-point variables, and the intrinsic engine state. Rather than relying on simplified empirical corrections for environmental conditions and operating-point variables, as in conventional physics-based models, this relationship is treated as a thermodynamic inverse problem. By learning the high-order nonlinear coupling between these operating conditions and gas path efficiency directly from historical operation, the proposed model functions as a physics-guided thermodynamic state observer. In this way, the influence of environmental and operating variations can be separated from the underlying performance behavior more reliably than in traditional analytical schemes, enabling a clearer identification of the intrinsic state. Although the intrinsic state determines the observed performance under identical environmental and operating conditions, it cannot be represented by a single measurable variable. Its evolution over time may also exhibit certain trends. If such latent states can be characterized and consistently applied across different engines, the resulting trend categories could provide meaningful abstractions of engine health evolution.

To this end, this paper proposes a Transferable Snippet Augmentation Network (TSAN) for predicting the degradation trends of aero-engines. The key idea is to leverage degradation trend representations that capture the underlying evolution patterns of engine states, enabling consistent characterization across different engines. By introducing the notion of degradation trend groups, the method facilitates the transfer of informative data snippets between engines with similar degradation periods. This allows the model to better generalize under varying operational and degradation conditions. Through this design, TSAN aims to enhance both the accuracy and the transferability of performance prediction in cross-engine scenarios. The main contributions of this paper are as follows:

1. A trend-grouped fine-tuning migration approach is proposed. Similar degradation-trend snippets are extracted from the historical data of all engines of the same type to fine-tune the basic prediction model. This design helps reduce the overall model scale required for engine maintenance in airlines.
2. For new engines with limited data, the method transfers data snippets with the same degradation trend group from other engines to enhance prediction accuracy. This strategy improves the applicability of the fine-tuned model to the target engine category.
3. The classification of engine degradation trends is visualized, showing that the learned trend divisions are of practical significance. In addition, the proposed method achieves superior prediction performance compared with existing approaches.

In the rest of this paper, the proposed method is first described in terms of its overall prediction method, model architecture, and training and testing procedures. The data acquisition and preprocessing process, together with the characteristics of the aviation engine dataset, are then presented to motivate the application of TSAN. Experimental settings, evaluation metrics, and result analyses are subsequently reported, followed by concluding remarks.

2. Methodology

This section systematically explained the principles and key processes of the TSAN method, focusing on two main aspects: first, it presented the overall process for predicting key performance parameters of aviation engines and provided a detailed description of the training and fine-tuning processes involved in the method's implementation; second,

it elaborated on the structural design concepts of TSAN.

2.1. Predictive methodology and workflow

To construct the dataset for model training and evaluation, flight data from each engine were processed using a sliding-window sampling strategy. Fig. 1(a) schematically illustrates the dataset construction process. The key performance parameters (yellow curves), control parameters, and environmental parameters (green curves) together constitute the set of variables on the vertical axis, while each timestamp on the horizontal axis represents the feature values corresponding to a specific stage within a flight. The flight data of each engine was first divided into training and testing sets, with the first 75% used for training and the remaining 25% for testing. The sliding window (red boxes in Fig. 1(a)) was then applied separately to the training and testing portions to generate samples, which served as training and testing instances, respectively. Finally, all training and testing samples were aggregated independently to form the final training and testing sets, which together constituted the experimental dataset.

The structure of each sample is illustrated in Fig. 1(b). The sampling length L represents the number of known flights included in each sample for the prediction task. After the $m-L$ to $m-1$ flights, the data from the m-th flight serves as the prediction target. In $\boldsymbol{x}_m$, the target variable for prediction is the *EGTM*, while the remaining parameters in the sample serve as covariates. During training, the *EGTM* values in $\boldsymbol{x}_m$ were initialized using the known data from the previous sample, i.e., the *EGTM* of $\boldsymbol{x}_{m-1}$.

Fig. 2 presents fine-tuning strategies from different perspectives, including the traditional single-engine personalized fine-tuning approach and the trend-based grouped fine-tuning approach. Specifically, Fig. 2(a) presents the conventional method for addressing engine-specific differences, as discussed earlier. On the left side of Fig. 2 (a), "All engines' data" refers to the complete training data from engines 1 to N. Through preprocessing, the data was first segmented into multiple fragments to train a basic model, referred to as the General Network via step ①. To predict the gas path parameters of engine n, its corresponding training samples were used to fine-tune the General Network in step ②, resulting in an Engine-Specific Network. Finally, in step ③, this Engine-Specific Network was used to predict the unknown parameters of engine n. Given that the proposed method required the incorporation of trend representation information into the model, RNN-based architectures were particularly suitable, as they naturally relied on an initial hidden state. Among them, the Gated Recurrent Unit (GRU) was adopted as the foundational model due to its efficient gating mechanism and proven effectiveness in engine-related prediction tasks [28]. To highlight the fine-tuning process based on an individual engine, this approach was referred to as Separate GRU.

Fig. 2 (b) shows the overall process of the TSAN-based engine-specific degradation trend prediction method proposed in this paper. This method consists of the following four steps:

Step 1: Train a basic TSAN for all engines

First, a basic model was constructed and trained using all samples from the training set. The training process focused on two main objectives: enhancing the model's ability to represent degradation trends and improving its predictive performance.

For the goal of representing degradation trends, the training objective was to integrate the representation module with the prediction module, enabling the degradation trend representations produced by the representation module to be effectively utilized by the prediction module. Compared with Fig. 1 (b), the input sequence ($\boldsymbol{x}_{m-2L}$ $\boldsymbol{x}_{m-1}$) was extended to twice its original length ($\boldsymbol{x}_{m-L}$ $\boldsymbol{x}_{m-1}$) and fed into the TSAN. The representation module took the input sequence $\boldsymbol{x}_{m-L}$ $\boldsymbol{x}_{m-1}$ and produced a degradation trend representation $\boldsymbol{h}_2$. Let H_s denote the size of the hidden state in the prediction module; then the dimension of $\boldsymbol{h}_2$ is $1 \times H_s$. The prediction module took the sequence $\boldsymbol{x}_{m-2L}$ $\boldsymbol{x}_{m-L-1}$ as input to obtain a hidden state $\boldsymbol{h}_1$, and the discrepancy between $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ was used as a loss function to optimize the representation module [29].

For the training process of the prediction module, the objective was to improve prediction accuracy. Specifically, by masking the *EGTM* values in each sample $\boldsymbol{x}_m$ from the training set, using the known sequences $\boldsymbol{x}_{m-L}$ $\boldsymbol{x}_{m-1}$ and the covariates in $\boldsymbol{x}_m$, the model was trained to predict the masked *EGTM*. The prediction discrepancy was then minimized using backpropagation to update network parameters. Then, the trained network was used to generate predictions $\tilde{y}$, which were then compared to the ground truth y to evaluate performance. In this paper, the method that employed only GRUs for prediction is termed Single GRU, and its training process constitutes a component of the TSAN.

During the training of the base network, the two modules were trained jointly, and the final loss function is formulated as:

$$\mathscr{L}_1 = \alpha \cdot \mathrm{MSE}(\boldsymbol{h}_1 - \boldsymbol{h}_2) + (1-\alpha) \cdot \mathrm{MSE}(y - \tilde{y}) \tag{1}$$

where α denotes a weighting coefficient that takes a value in the range [0, 1].

To formalize the relationship between $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$, the degradation

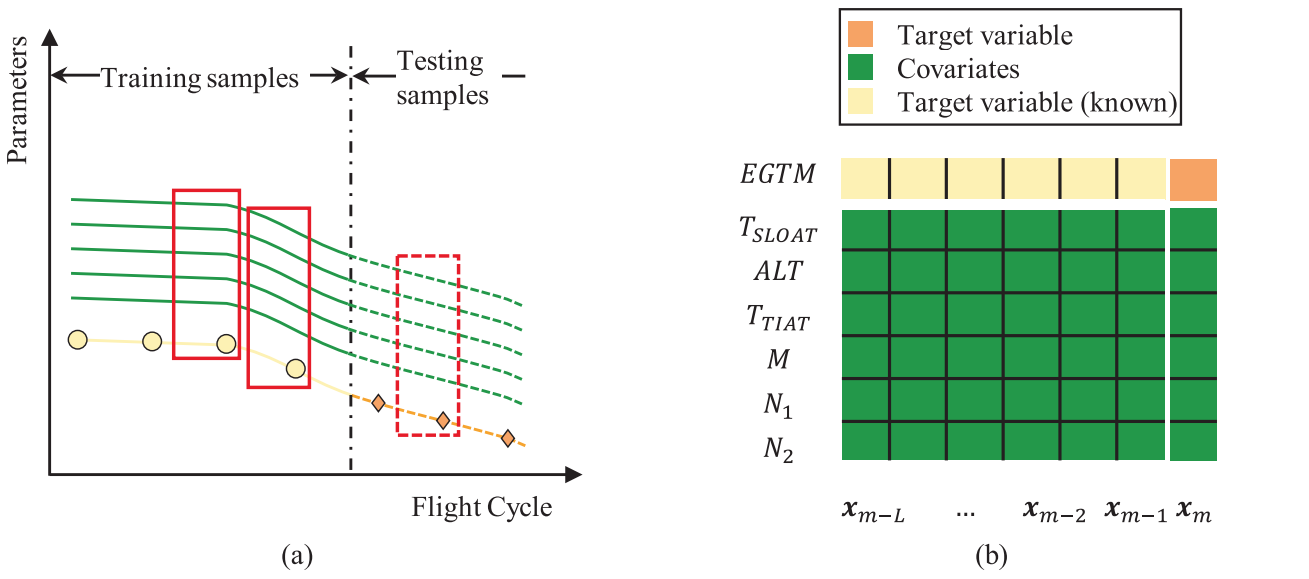

Fig. 1. Schematic illustration of (a) the dataset construction process and (b) the input sample structure for the key performance parameter prediction task.

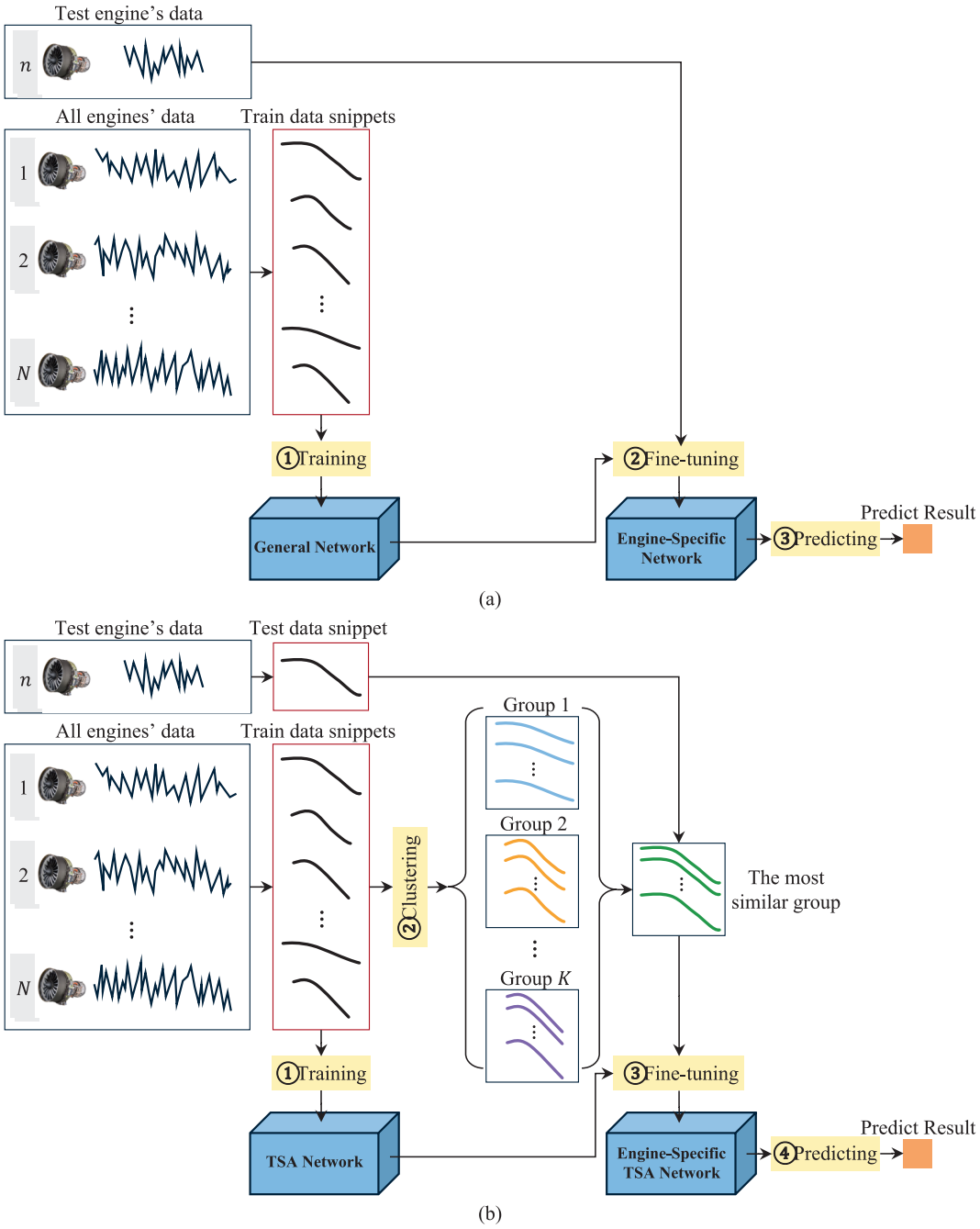

Fig. 2. Different fine-tuning strategies: (a) Traditional single-engine personalized fine-tuning process; (b) Trend-based grouped fine-tuning process.

process is represented as a latent continuous dynamical system with hidden states $\boldsymbol{S}_m$. Given the physical continuity and state uniqueness of the degradation dynamics, the latent state at the boundary index L must be consistent, regardless of the direction of inference. Therefore, $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ can be viewed as two conditionally independent estimators of the same latent state $\boldsymbol{S}_L$:

$$\boldsymbol{h}_1 \approx \mathbb{E}[\boldsymbol{S}_L|\boldsymbol{x}_{0:L}], \boldsymbol{h}_2 \approx \mathbb{E}[\boldsymbol{S}_L|\boldsymbol{x}_{L:2L}] \tag{2}$$

These two quantities constitute conditionally independent estimators of the same ground-truth variable derived from different observation windows. Any discrepancy between them is attributed to estimation noise rather than physical divergence. This deviation is modeled as:

$$\boldsymbol{h}_1 = \boldsymbol{h}_2 + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2 I) \tag{3}$$

The corresponding likelihood function is:

$$p(\boldsymbol{h}_1|\boldsymbol{h}_2) \propto \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{h}_1 - \boldsymbol{h}_2\|_2^2\right) \tag{4}$$

Maximizing this likelihood is mathematically equivalent to minimizing the negative log-likelihood, which yields the squared Euclidean penalty term: $\|\boldsymbol{h}_1 - \boldsymbol{h}_2\|_2^2$. Thus, the MSE alignment term is grounded in the Maximum Likelihood Estimation principle for state consistency, rather than being a heuristic regularization term. This ensures that $\boldsymbol{h}_2$ remains aligned with $\boldsymbol{h}_1$.

Step 2: Clustering of snippets with similar degradation trends

After the basic model was trained, each snippet in the training set was assigned a degradation trend group based on the output of the representation module. Using a sliding window of length L, both the

training and test samples were reprocessed through the representation module to generate a new set of training samples. Let T denote the number of new training samples, denoted as $\boldsymbol{X}_c$, which were fed into the degradation trend representation module. The module outputs a trend representation for each sample, denoted as $\boldsymbol{H}_c$, with the same structure as $\boldsymbol{h}_2$. The shape of $\boldsymbol{H}_c$ is $T \times H_s$.

After obtaining the trend representation, each sample must be assigned to a corresponding degradation trend group. Several clustering methods were evaluated, including K-Means, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Ordering Points to Identify the Clustering Structure (OPTICS), hierarchical clustering, and Gaussian Mixture Model (GMM). The final choice was determined by the structural requirements of degradation modeling. The latent representation exhibits a gradual and continuous evolution, which necessitates a clustering model capable of providing probabilistic memberships and maintaining stable, transferable assignment rules. In addition, the clustering structure must support efficient inference for new flight cycles without relying on dataset-specific density parameters or repeated neighborhood searches. GMM inherently satisfies these conditions through a parametric likelihood formulation with fixed-size parameters, enabling soft memberships and constant-time assignment. In contrast, nonparametric or hard-partitioning methods do not inherently provide these capabilities. For these reasons, GMM was adopted as the clustering method in this study. The process is defined as:

$$\boldsymbol{l} = \mathrm{GMM}(\boldsymbol{H}_c, K) \tag{5}$$

where $\boldsymbol{l}$ is the vector of cluster labels for the T samples, and K is the number of clusters used in GMM. The function $\mathrm{GMM}(\cdot,\cdot)$ denotes the standard GMM clustering algorithm. In this paper, the number of clusters K is adaptively determined based on the Bayesian Information Criterion (BIC) following the principle of marginal improvement. To ensure sufficient model fitting quality while avoiding an excessive number of clusters that would lead to insufficient data within each group, an improvement ratio threshold τ is first defined. The BIC improvement ratio between two consecutive values of K is formulated as

$$\rho(K) = \frac{\mathrm{BIC}(K) - \mathrm{BIC}(K+1)}{\mathrm{BIC}(K-1) - \mathrm{BIC}(K)} \tag{6}$$

where $\mathrm{BIC}(\cdot)$ denotes the BIC value corresponding to a given number of clusters K. When the improvement ratio $\rho(K) < \tau$, it indicates that further increasing model complexity yields only marginal reductions in BIC. In such cases, adding more clusters becomes unnecessary and may result in insufficient transferable data within each group.

Through this process, the degradation trend representation of each sample was classified into one of K classes, allowing each sample to be associated with a specific class that reflects its underlying engine degradation trend. If the categorization is meaningful, then at the level of a single engine, the assigned groups should evolve over flight cycles in a manner that simplifies into a few interpretable degradation patterns.

Step 3: Fine-tuning based on transferable snippet augmentation

Based on the degradation trend group presented by the test sample, a large number of similar samples can be retrieved from the training set. After obtaining the test sample, it was first passed through the representation module to generate a feature representation. This representation was then compared with those of engine data snippets in the training set to determine the degradation trend group to which the test sample belongs. Once the group was identified, the training samples most similar to the test sample (referred to as the most similar group in Fig. 2(b)) are determined. All samples within this group were then used to fine-tune the TSAN, yielding the Engine-Specific TSAN.

During the initial training of the basic TSAN, the prediction module's initial hidden state was generated by the representation module. In the fine-tuning phase, both the degradation trend representation module and the prediction module were jointly trained to enhance the predictive performance of TSAN. The loss function is defined as follows:

$$\mathscr{L}_2 = \mathrm{MSE}(y - \tilde{y}) \tag{7}$$

Step 4: Differentiated degradation trend prediction based on TSAN

The trained Engine-Specific TSAN was used for the prediction task on test samples. First, the test data was segmented into samples using a sliding window, with each sample having a length of $L+1$ along the flight dimension, which was consistent with the input format of traditional neural network-based prediction methods. Next, the degradation trend group corresponding to each test sample was determined. This was done by generating a degradation trend representation using the representation module trained in the basic model. The representation was clustered using a GMM, and once the trend group membership was identified, the fine-tuned TSAN corresponding to that trend group was used for prediction.

These steps effectively reduced the number of test engine samples required by leveraging the degradation trend group as the primary basis for model fine-tuning, thereby making full use of existing data from engines of the same type. The following section provides more detail on key modules of the overall process, including the basic TSAN architecture and the fine-tuned model structure.

2.2. Design of TSAN network structure

Fig. 3 shows the structure of the TSAN. For the time series data to be predicted $\boldsymbol{x}_m$, the input of a traditional prediction network is typically $\{\boldsymbol{x}_{m-L}, \boldsymbol{x}_{m-L+1}, \cdots, \boldsymbol{x}_m\}$, while the input to TSAN extends the length of the known data sliding window by a factor of two, expressed as $\{\boldsymbol{x}_{m-2L}, \boldsymbol{x}_{m-2L+1}, \cdots, \boldsymbol{x}_m\}$. For clarity, the data within the red solid wireframe area, $\{\boldsymbol{x}_{m-2L}, \boldsymbol{x}_{m-2L+1}, \cdots, \boldsymbol{x}_{m-L-1}\}$, is denoted as $\boldsymbol{X}_l$, and the data within the blue solid wireframe area is denoted as $\boldsymbol{X}_r$.

Unlike the General Network structure of Separate GRU, TSAN includes two modules: the representation module and the prediction module. The black box on the left side of Fig. 3 shows the main steps of the representation module in TSAN. This module outputs a set of data that effectively represents the degradation trend of the engine within the given sample. In the GRU model, the initial hidden state is typically set to a zero vector, and data from different time steps are sequentially fed into the GRU to update the hidden state variables. The final hidden state can then effectively express the overall information of the input sample. However, the GRU updates the hidden state progressively through forward propagation. To effectively represent the GRU's initial hidden state, additional networks are needed to integrate this information. Moreover, considering that data from different flight stages within the same flight have varying significance in revealing the engine's degradation trend, the importance of these stages differs. For example, during takeoff, the engine's sensor data more accurately represents its operating state due to the higher load, whereas during cruise, the data is less representative of the engine's condition as the engine is not under significant stress. Therefore, in a data sample containing multiple flights, the data from different flights provides varying levels of insight into the degradation trend. To better represent the degradation trend of the samples, this paper designs the representation module of the TSAN, which integrates the importance of data from different flights within the same sample. First, the data $\boldsymbol{X}_r$ is input into two layers of Transformer encoders to obtain the transformed features, expressed as

$$\boldsymbol{Z} = \mathrm{TransformerEncoder}(\boldsymbol{X}_r) \tag{8}$$

The shape of $\boldsymbol{Z}$ is the same as that of $\boldsymbol{X}_r$. The transformed feature $\boldsymbol{Z}$ is passed through a linear layer to generate a sequence of hidden variables $\boldsymbol{H}$. Then, $\boldsymbol{Z}$ is passed through another linear layer, followed by a Softmax function to compute the attention weights $\boldsymbol{w}$ corresponding to each time step in $\boldsymbol{H}$. This process is expressed as:

$$\boldsymbol{H} = \mathrm{Linear}(\boldsymbol{Z}) \tag{9}$$

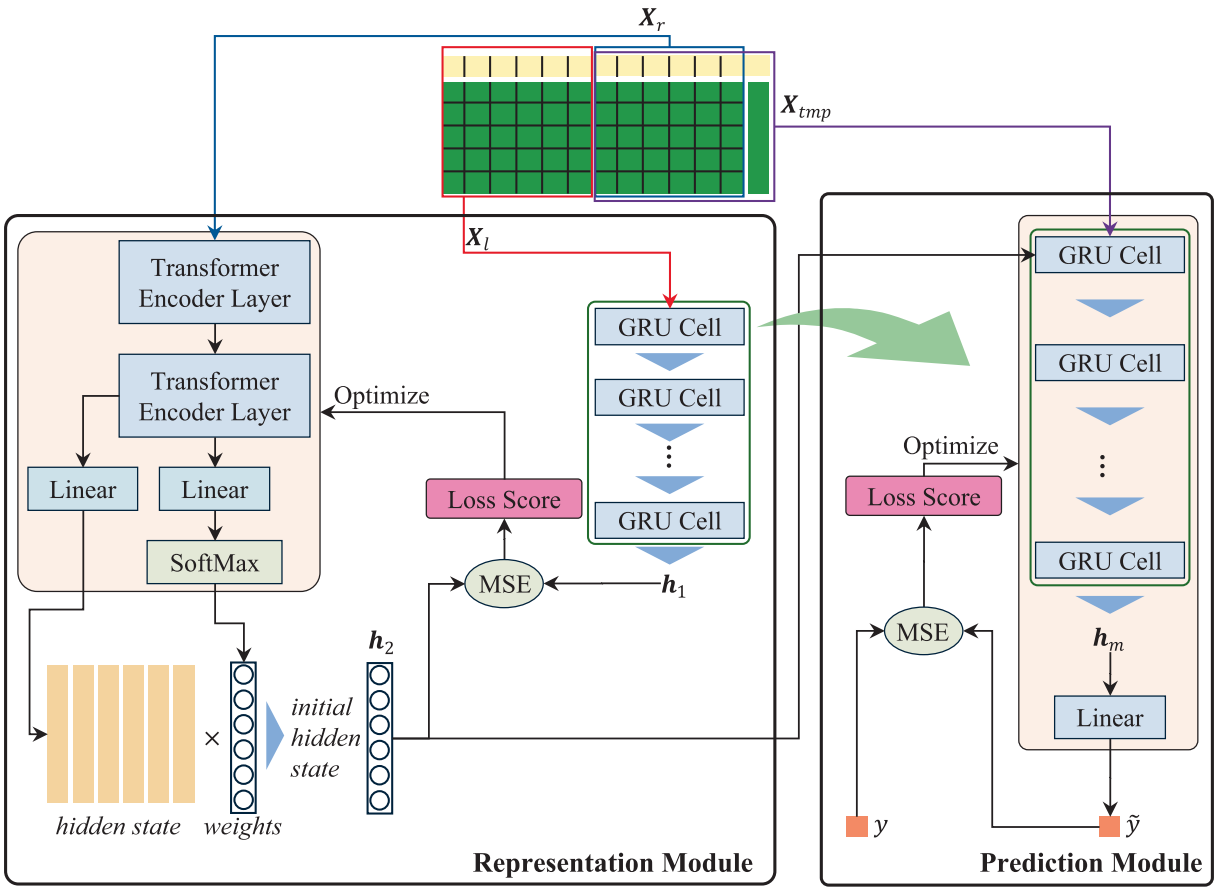

Fig. 3. Structure of the TSAN.

$$\boldsymbol{w} = \text{Softmax}(\text{Linear}(\boldsymbol{Z})) \tag{10}$$

Here, $\boldsymbol{H}$ denotes the hidden representations in the yellow area at the bottom left of Fig. 3, with shape $L \times H_s$, where H_s is the size of each hidden vector. The attention weight vector $\boldsymbol{w}$ has a shape of $1 \times L$, assigning importance to each time step. To obtain a hidden representation that best captures the state of $\boldsymbol{X}_r$, a weighted sum of $\boldsymbol{H}$ is computed using $\boldsymbol{w}$, resulting in vector $\boldsymbol{h}_2$, which serves as the initial state for subsequent prediction:

$$\boldsymbol{h}_2 = \boldsymbol{w} \times \boldsymbol{H} \tag{11}$$

Next, the time series data $\boldsymbol{X}_l$ (red solid box) is input into a GRU prediction network identical to that used in the Single GRU method. The initial hidden state is set to a zero vector, and the GRU processes $\boldsymbol{X}_l$ to produce a hidden representation $\boldsymbol{h}_1$:

$$\boldsymbol{h}_1 = \text{GRU}(\boldsymbol{X}_l, \boldsymbol{z}) \tag{12}$$

where $\boldsymbol{z}$ denotes a zero vector, and $\text{GRU}(\boldsymbol{X}_l, \boldsymbol{z})$ represents the gated recurrent unit operation.

The right black box in Fig. 3 shows the main steps of the prediction module in TSAN. First, for prediction, the model must predict unknown parameters. To this end, $\boldsymbol{X}_r$ is concatenated with $\boldsymbol{x}_m$ along the time dimension to form the prediction input $\boldsymbol{X}_{tmp}$ (the purple solid box in Fig. 3):

$$\boldsymbol{X}_{tmp} = \text{Concat}(\boldsymbol{X}_r, \boldsymbol{x}_m) \tag{13}$$

where $\text{Concat}(\boldsymbol{X}_r, \boldsymbol{x}_m)$ represents the concatenation operation along the time dimension. Then, the GRU used in the previous step is reused as the prediction network, but with its initial hidden state set to $\boldsymbol{h}_2$. The GRU processes $\boldsymbol{X}_{tmp}$ to produce the hidden representation $\boldsymbol{h}_m$ for the time-stamp m:

$$\boldsymbol{h}_m = \text{GRU}(\boldsymbol{X}_{tmp}, \boldsymbol{h}_2) \tag{14}$$

Finally, $\boldsymbol{h}_m$ is passed through a linear layer to reduce its dimensionality from H_s to 1, producing the final prediction result $\tilde{y}$:

$$\tilde{y} = \text{Linear}(\boldsymbol{h}_m) \tag{15}$$

Fig. 4 shows the structure of the fine-tuning and testing process of TSAN. After the basic network is trained, its representation module is retained to generate degradation trend representations for test samples. For each group-specific task, the representation and prediction modules are jointly optimized during fine-tuning and testing. The optimization uses backpropagation based on the difference between predicted and actual values. This process improves prediction accuracy within each group.

3. Experimental results

This section was organized into three main parts: data acquisition and analysis, experimental design, and analysis of experimental results. First, the data acquisition process and key parameters were introduced, followed by an analysis of the feasibility of the TSAN method. Next, the main hyperparameter settings and difference measurement methods used in the experiments were presented, with certain network architecture parameters determined via grid search. Third, experiments were conducted from both the engine-level and trend-level perspectives, and the differences in results between these two approaches were analyzed. Finally, several baseline methods were applied to the same prediction task, and the results were compared and analyzed from multiple perspectives to comprehensively evaluate the performance of the proposed method.

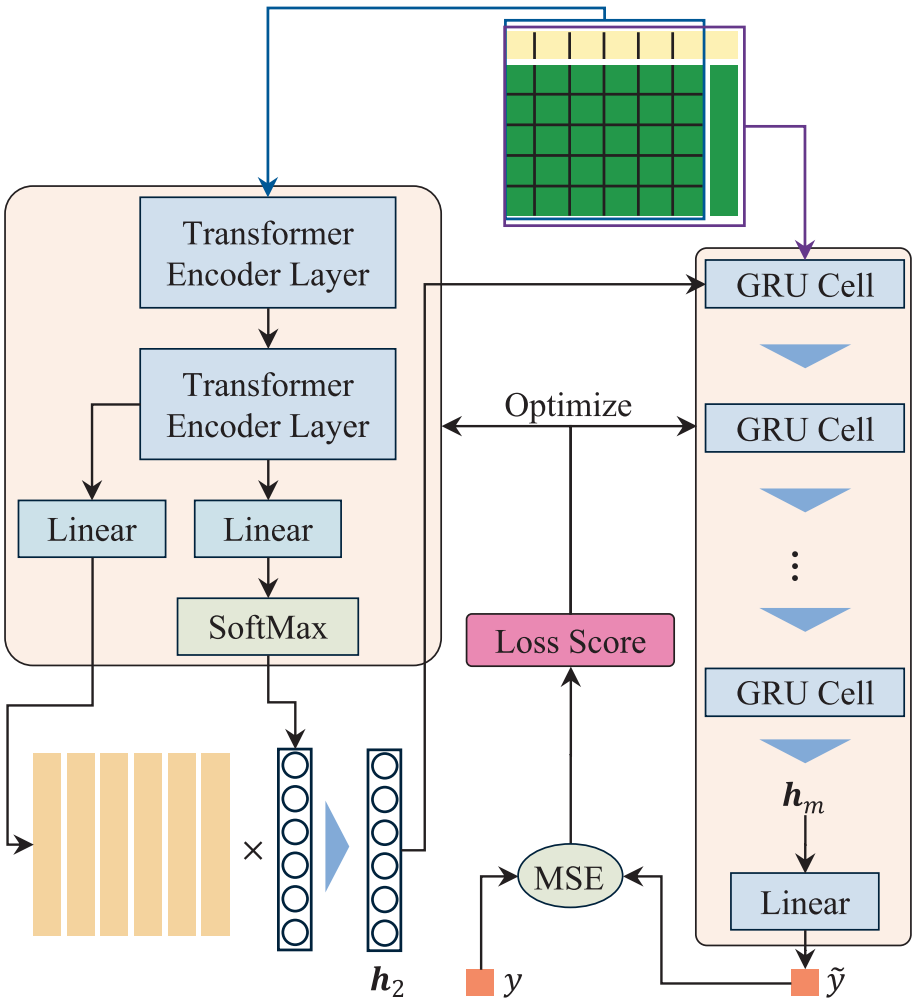

Fig. 4. The structure of the fine-tuning and testing process of TSAN.

3.1. Data acquisition, analysis, and preprocessing

This paper focuses on a specific type of engine and analyzed data collected during the takeoff phase of multiple consecutive flights that had not undergone an overhaul. To avoid introducing additional errors caused by missing data, the experimental dataset was constructed based on 18 engines with complete data from a commercial airline. The original data include Aircraft Communications Addressing and Reporting System (ACARS) messages recorded during the takeoff phase, as well as relevant supporting information provided by the original equipment manufacturer (OEM). The boundary conditions of the numerical procedure are defined from two aspects. First, the valid domain represents the physical operating envelope restricted to the Take-off and Climb phases. The valid operational domain of the model is defined by the parameter ranges captured within the historical training data. Second, the numerical boundary is determined by the statistical distribution of the dataset. A standardization scaler is fitted to learn the mean and variance of the training samples. This scaler is applied to normalize inputs, and any values deviating significantly from the statistical bounds of the training distribution are considered outside the valid boundary of the numerical procedure.

The prediction task involves seven relevant parameters. The target of this experiment is the smoothed Exhaust Gas Temperature Margin (EGTM), a value provided by the OEM. After production or overhaul, the OEM establishes a performance baseline for each engine and, by combining flight data (e.g., ACARS reports) with performance correction models, calculates the EGTM. A larger EGTM indicates that the engine retains better performance and operates further from the exhaust gas temperature limit, whereas a smaller EGTM reflects performance deterioration and implies that the engine is approaching its operational limit. The covariates include six parameters: Sea Level Outside Air Temperature (T_{SLOAT}), Altitude (ALT), Total Inlet Air Temperature (T_{TIAT}), Mach Number (M), Fan Speed (N_1), and Core Engine Speed (N_2).

Under the assumption of quasi-steady-state operation, the selected covariates sufficiently capture the core thermodynamic relationships of the Brayton cycle. The combination of environmental variables (T_{SLOAT}, ALT, T_{TIAT}), flight condition (M), and engine operating-point variables (N_1, N_2) jointly characterizes the inlet boundary conditions, compressor loading, and turbine work balance that determine the steady-state gas path. The criticality of each parameter depends on its physical meaning within the gas path. Among all inputs, N_2 and N_1 are the most critical because they directly characterize the work balance of the core and fan modules and are therefore highly sensitive to reductions in compressor or turbine efficiency. In contrast, T_{TIAT}, M, and ALT primarily describe the external operating conditions that define the thermodynamic boundaries of the cycle, and these variables determine the engine's operating point. T_{SLOAT} is identified as the least critical among the covariates because its thermodynamic information is largely embedded in T_{TIAT}, while it remains necessary as an ambient reference that stabilizes cross-mission environmental variability. Constraining the learning process with these physically meaningful variables ensures that variations in the predicted target parameter reflect actual component health changes rather than artifacts arising from shifts in operating conditions. Mathematically, these covariates restrict the network to learning within

the physically admissible operating region defined by the engine gas path governing equations. For the n-th engine that has completed C flights, the data from the m-th flight is represented as $x_m = \{EGTM, T_{SLOAT}, ALT, T_{TIAT}, M, N_1, N_2\}$ and the input for the prediction task is represented as $X_n = \{x_1, x_2, \cdots, x_M\}$. In addition to the above parameters, this paper also collected the aircraft ID, engine position, engine serial number, and flight timestamp to support data differentiation, chronological ordering, and hypothesis validation. Specifically, the engine serial number is used to distinguish different engines, the flight timestamp is used to determine the sequence of data from the same engine.

An initial analysis of the engine data used in the experiment was conducted from the perspectives of data volume, data distribution, and data differences. The results are shown in Table 2. In Table 2, No. indicates the serial number of the engine analyzed, and Cycle refers to the number of flight cycles used for each engine (i.e., the value of C mentioned earlier). For data distribution, the Mean and Variance represent the average and variability of EGTM values for each engine, respectively, while Outliers indicate the number of data points that deviate from the EGTM distribution based on the 3 σ rule. For data differences, MAE, RMSE, and Range denote the mean absolute error, root mean square error, and range of the differences between the raw and smoothed EGTM values, respectively. From Table 2, it can be observed that engines No. 12 and No. 18 have the largest data volumes, each with 1949 flight records. Engines No. 3 and No. 17 have the fewest data, with only 124 and 139 flights, respectively. The largest variances in EGTM are observed for engines No. 11 and No. 12, at 93.28913 and 92.63143, respectively. Engine No. 10 contains the highest number of outliers, with a total of 9. In terms of the difference between raw and smoothed EGTM, engines No. 1 and No. 17 exhibit the highest MAE, engines No. 16 and No. 17 have the highest RMSE, and engines No. 12 and No. 18 show the largest ranges.

For engine key performance prediction tasks, the basic principle is to construct samples using adjacent data and covariates to estimate the target performance parameters. In this paper, the internal parameter transformation process of the engine is simplified as a function $f(\cdot)$, where the input consists of environmental conditions and control parameters. This process can be represented by the following function:

$$EGTM = f(T_{SLOAT}, ALT, T_{TIAT}, M, N_1, N_2, \theta) \tag{16}$$

The first six input parameters are covariates, while θ represents the engine state inferred from other flights within the same sample. However, different flights in a sample may vary in how well they represent the engine's operational state. For instance, some flights only experience stable operating conditions and may not fully engage the engine's performance, making them less informative for state estimation. In contrast, certain flights may exhibit data patterns that more accurately reflect the engine's true condition [30]. Before designing a trend-based prediction model, it is essential to first analyze the general patterns of engine data across different operational stages. This analysis involves both cross-engine comparisons of similar stages and intra-engine comparisons across different stages. In the case of different engines, identifying similar operational stages suggests that certain patterns of engine behavior are consistent across engines and can be grouped for deeper exploration. Within the same engine, if data patterns differ significantly across stages, this indicates that modeling each stage separately may be more appropriate. The method proposed in this paper thus has the potential to yield more accurate predictions by accounting for these distinctions.

Fig. 5 shows the EGTM variation processes of the 13th engine (denoted as ②), which has fewer flight cycles, and the 15th engine (denoted as ①), which has more flight cycles. For ease of comparison, the smoothed EGTM data provided by the OEM is labeled as EGTM_S. Dynamic time warping (DTW) is used to measure the similarity between the overall EGTM_S sequence of the 13th engine and segments from the 15th engine. After normalization, the two segments are represented as $\boldsymbol{s}_1, \boldsymbol{s}_2$. The similarity distance is computed as $D = D_{DTW}(\boldsymbol{s}_1, \boldsymbol{s}_2)/\widetilde{L}$, where $\widetilde{L}$ denotes the average length of the two segments, and $D_{DTW}(\cdot)$ is the DTW distance. The similarity score is then transformed into the interval $[0, 1]$ using $S = e^{-D}$. The results indicated that a segment of the smoothed EGTM sequence starting from the 1042nd flight cycle of the 15th engine shows a strong similarity to that of ②. According to the DTW-based similarity calculation, the two different segments achieved a similarity of 0.7024, indicating that their overall trends are similar. By extracting and plotting the corresponding EGTM_S segments of ② and ①, the resulting curve (denoted as ③) reveals a certain degree of similarity in the predictor variable across different engines, under the assumption that covariates are ignored.

Taking the 15th engine as the subject for analyzing different operational stages of the same engine, ④, ⑤, and ⑥ in Fig. 5 show the EGTM and the smoothed EGTM variation processes of three segments, each consisting of 60 consecutive flight cycles. From the smoothed EGTM curves, it can be observed that although the overall EGTM exhibits a decreasing trend as the number of flight cycles increases, the rate and pattern of decline differ across stages. Through DTW-based similarity analysis, the similarity values between segments ④ and ⑤, ④ and ⑥, and ⑤ and ⑥ were 0.4612, 0.2927, and 0.3904, respectively, indicating relatively weak similarity. This observation suggests that while different engines may exhibit similar degradation patterns during certain usage

Table 2
Data quality assessment of the engine flight records.

No.	Cycles	Distribution			Differences		
		Mean	Variance	Outliers	MAE	RMSE	Range
1	443	68.68447	41.64013	4	3.49704	4.45658	23.81290
2	343	49.68660	41.15185	2	2.49620	3.61309	31.61390
3	124	53.14160	22.58389	1	2.97301	3.77809	12.36100
4	842	97.43781	25.19950	3	2.38380	3.27426	28.25040
5	378	57.80721	61.44141	4	2.65875	4.08617	30.80760
6	205	35.00245	70.81926	0	2.86774	3.96546	18.10700
7	214	80.89153	42.60896	0	3.18460	4.53795	21.47920
8	957	68.53910	63.63548	1	2.71414	3.55240	26.57700
9	162	64.28829	14.24542	1	2.14122	2.78944	12.95200
10	1716	68.10182	69.96648	9	2.63775	3.47804	35.42710
11	1716	69.88849	93.28319	2	2.55778	3.35166	33.03610
12	1949	62.15226	92.63143	3	2.72129	4.16601	88.81550
13	479	46.68781	35.97099	4	2.49589	3.14981	10.41510
14	722	57.76647	45.11239	4	2.65529	3.39978	13.73480
15	1566	59.18808	65.95846	2	2.43007	3.06992	14.87840
16	381	65.15077	59.99903	4	3.09803	4.89189	43.11770
17	139	79.06602	57.97307	0	3.48187	4.61635	20.99880
18	1949	68.03994	70.26179	4	2.70266	4.27079	91.05300

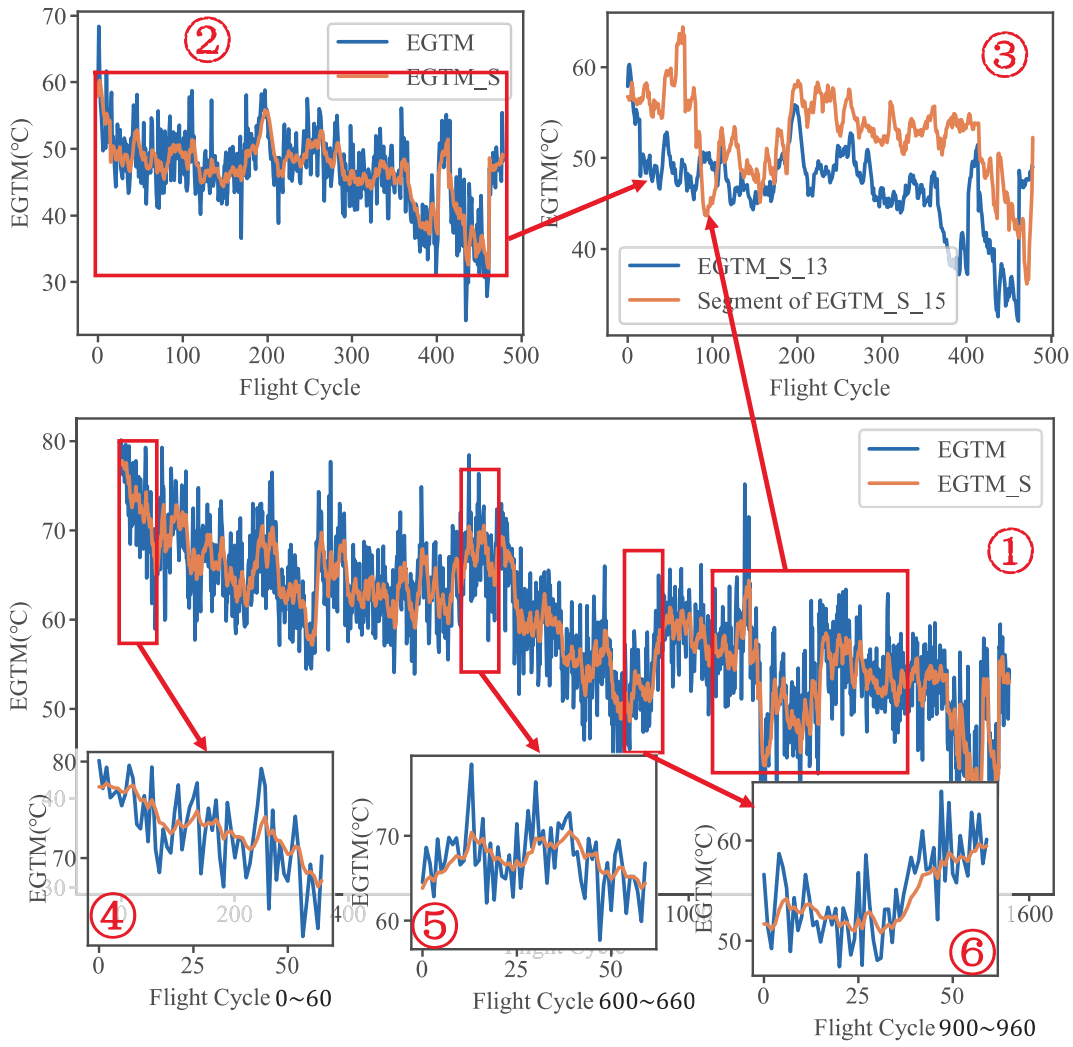

Fig. 5. Comparison of EGTM change processes: multiple engines vs. different stages of a single engine.

phases, distinct stages of the same engine can display divergent performance evolution. Therefore, the approach proposed in this paper is feasible for predicting key engine parameters.

Before being fed into the neural network, the raw data underwent a series of preprocessing procedures, including data partitioning, standardization, and sliding window segmentation. The detailed process is described as follows:

a) The dataset was divided according to engine identifiers, and the data corresponding to each individual engine were arranged in chronological order.
b) For each engine, a certain proportion of the time-series data was allocated to the training set, with the remaining portion reserved for testing. This ensured a strict separation between the training and test sets and effectively prevented information leakage.
c) All training data were then aggregated, and a scaler was fitted based on their statistical properties to learn the parameters required for standardization.
d) The fitted scaler, derived from the training data distribution, was subsequently applied to both the training and test sets to ensure consistent data scaling.
e) Finally, the sliding window technique was employed separately on the training and test data of each engine to construct the final input datasets for the neural network.

3.2. Experimental setups

To minimize the impact of hyperparameter differences among the compared models, this paper adopted identical parameter settings for the shared components across different networks in the comparison experiments. The detailed hyperparameter configurations are categorized as follows:

a) Network architecture.

Specifically, once the GRU hyperparameters were determined, these settings were consistently applied to all subsequent structures involving the GRU module. Based on the hyperparameter tuning results, the GRU network was configured with 32 hidden units and 1 layer.

b) Training strategy.

The models were trained using the Adam optimizer with an initial learning rate of 0.0001. The batch size was uniformly set to 512, and the maximum number of training epochs was set to 1600. To prevent overfitting and reduce unnecessary training, an early stopping strategy was employed, where training was terminated if the validation loss did not decrease for 20 consecutive epochs.

c) Data partitioning and prediction setup.

The prediction length L was consistently set to 10. For sequential prediction, a sliding window approach with a single-step stride was utilized. The size of the sliding window was set to $2L + 1$, i.e., 21.

d) GMM clustering configuration

In the GMM clustering module, numerical stability in the high-

dimensional space was ensured by restricting the covariance matrix to be diagonal, with a regularization term of 10^{-4} added to the diagonal elements. The convergence threshold was set to 10^{-3}, with the maximum number of iterations limited to 200. To avoid local optima, the number of random initializations was set to 10. The number of clusters K was varied from 1 to 20, and the parameter τ was set to 0.5.

e) Evaluation protocol.

For robust performance evaluation, each method was independently tested 10 times, and the average results were reported.

The average prediction from 10 repeated experiments is used as the final prediction result of each method, denoted as $\widetilde{\boldsymbol{y}}$. The error between $\widetilde{\boldsymbol{y}}$ and the ground truth $\boldsymbol{y}$ is evaluated using three metrics: MAE, MRE, and RMSE, which are calculated as follows:

$$\mathrm{MAE}(\boldsymbol{y},\widetilde{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widetilde{y}_i| \tag{17}$$

$$\mathrm{MRE}(\mathbf{y},\widetilde{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \widetilde{y}_i}{y_i}\right| \tag{18}$$

$$\mathrm{RMSE}(\mathbf{y},\widetilde{\mathbf{y}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widetilde{y}_i)^2} \tag{19}$$

Here, i denotes the index of an element in the vector, e.g., y_i refers to the i-th element of the vector $\boldsymbol{y}$; n represents the total number of elements in the predicted and actual value vectors.

3.3. Benchmark experiments using GRU

To refine the benchmark model, repeated experiments were conducted with different combinations of hidden size, number of layers, and learning rate. The optimal hyperparameter configuration was determined based on the observed performance.

Since TSAN is built upon a GRU-based structure in the prediction module, this section begins with experiments using a single GRU model. A grid search was conducted over various hyperparameter combinations, and each configuration was repeated 10 times. The evaluation was based on the average MAE, MRE, and RMSE metrics across 18 engines to determine the optimal hyperparameter setting. The main hyperparameters of the GRU model include the hidden layer size and the number of layers, while the key training parameters include the learning rate and the number of training epochs. Specifically, the number of layers was set to 1 or 2, the hidden layer size to 32 or 64, and the learning rate was chosen from 0.01, 0.001, or 0.0001. As shown in Table 3, the best performance among all combinations was achieved when the number of layers was 1, the hidden size was 32, and the learning rate was 0.0001. Under this configuration, the model attained an average MAE of 0.02895, MRE of 0.00057, and RMSE of 0.04023.

Fig. 6(a) shows the change in the loss during the training process of the first experiment using this method. As shown in the figure, training terminated early before reaching 1250 epochs, indicating that the network was able to converge effectively. Table 4 presents the experimental results for each of the 18 engines under the optimal hyperparameter settings identified through the grid search. As shown in the table, the performance of this method varies considerably across different engines, indicating relatively high volatility. To investigate the relationship between the amount of engine data and the imputation performance, engines were sorted in descending order based on data volume. Fig. 6(b) displays the ranking of performance metrics as engine data decreases. Along the x-axis of the figure, engine data volume decreases from left to right. It can be observed that the engines on the left side of the figure generally rank higher in performance, while those on the right side tend to rank lower. Among them, the 18th, 10th, and 11th engines with larger data volumes achieve the best results, whereas the 6th, 2nd, and 3rd engines with relatively less data show poorer performance.

Table 3
Performance of GRU benchmark with varying hyperparameters.

Layer	Hidden size	Learning rate	MAE	MRE	RMSE
1	32	0.0001	0.02895	0.00057	0.04023
1	32	0.001	0.03587	0.00067	0.04762
1	32	0.01	0.05314	0.00107	0.06901
1	64	0.0001	0.03381	0.00065	0.04652
1	64	0.001	0.04276	0.00084	0.06111
1	64	0.01	0.05755	0.00113	0.07433
2	32	0.0001	0.03546	0.00068	0.05025
2	32	0.001	0.04419	0.00087	0.06353
2	32	0.01	0.06597	0.00129	0.09182
2	64	0.0001	0.04803	0.00093	0.06805
2	64	0.001	0.06355	0.00129	0.08918
2	64	0.01	0.05462	0.00103	0.08102

3.4. Comparison of results from the engine and trend perspectives

To address the significant variation in operating conditions across different engines, Separate GRU was adopted, in which an individual GRU model was created for each engine. Initially, a base network was trained using data from all engines. Subsequently, each model was fine-tuned using the corresponding engine's individual data. The fine-tuned models were then used to predict the respective engines. The results are presented in Table 5.

The experimental results of Single GRU were compared with those of Separate GRU. Although the average results across the three evaluation metrics show only minor differences between the two approaches, a more detailed comparison from the perspective of individual engines reveals distinct patterns. When analyzing the MAE metric of the six engines with relatively smaller data volumes, it can be observed that although Separate GRU outperforms Single GRU on certain engines (e.g., Engines No. 3, 6, and 17), the overall improvement of Separate GRU among these six engines is only 0.98% on average. Similar trends are observed for the other evaluation metrics as well. This shows that for engines with limited data, fine-tuning provides only marginal performance gains. However, Separate GRU exhibits a significant reduction in the standard deviations of all three metrics across all 18 engines, indicating that this method yields more stable and consistent results.

Next, experiments were conducted from the perspective of trend representation. The proposed TSAN method was used to predict key gas path performance parameters of aviation engines. First, since the hyperparameter α is introduced in the trend representation module to control the parameter setting of the trend representation module, a grid search was conducted to determine the optimal value of this hyperparameter. In this study, α was varied from 0 to 1 with an interval of 0.2, and each configuration was evaluated through 10 repeated experiments. As shown in Table 6, the optimal value of α is 0.8, under which the averaged results of MAE, MRE, and RMSE across the repeated experiments are 0.01853, 0.00036, and 0.02651, respectively. All three metrics achieved their best performance at this setting. In addition, Table 6 also shows that when α is 0 or 1, the results are worse than when the intermediate value is taken. This indicates that the network achieves better results only when the two parts of the trend representation module work together.

For the first experiment, Fig. 7 shows the BIC values for different numbers of clusters K, and the optimal number of clusters in this experiment is 7. To quantitatively assess the classification quality, the posterior probability for each sample in the test set was calculated based on the optimal K value derived from the GMM. In an unsupervised context, the mean of these maximum posterior probabilities serves as the metric for classification accuracy, while its complement represents the margin of error. Across 10 independent experiments, an average clustering confidence of 99.13% was achieved, corresponding to a margin of error of 0.87%. These results indicate that the engine degradation trends are distinct and the clusters are highly separable, confirming the

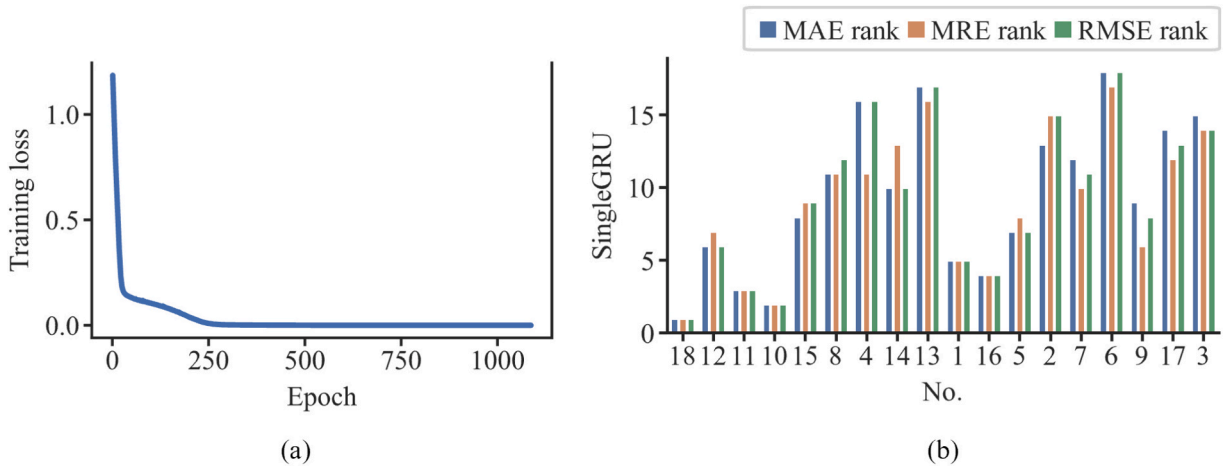

Fig. 6. Experimental processes and results of Single GRU. (a) Relationship between loss and epoch during training; (b) Relationship between engine data volume and prediction performance ranking.

Table 4
Experimental results of Single GRU.

No.	Total cycles	Testing cycles	MAE	MRE	RMSE
1	443	111	0.01279	0.00020	0.01822
2	343	86	0.03331	0.00084	0.04633
3	124	31	0.03753	0.00076	0.04462
4	842	211	0.04021	0.00043	0.05752
5	378	95	0.01394	0.00028	0.01917
6	205	52	0.12181	0.00294	0.19694
7	214	54	0.03168	0.00041	0.03856
8	957	240	0.02862	0.00043	0.03933
9	162	41	0.01594	0.00025	0.01964
10	1716	429	0.00959	0.00016	0.01252
11	1716	429	0.01003	0.00017	0.01323
12	1949	488	0.01382	0.00027	0.01883
13	479	120	0.05608	0.00152	0.06985
14	722	181	0.02388	0.00052	0.03851
15	1566	392	0.01549	0.00032	0.02337
16	381	96	0.01086	0.00018	0.01368
17	139	35	0.03675	0.00046	0.04163
18	1949	488	0.00873	0.00014	0.01225
AVG ± STD	\	\	0.02895 ± 0.02680	0.00057 ± 0.00068	0.04023 ± 0.04262

Table 5
Experimental results of Separate GRU.

No.	Total cycles	Testing cycles	MAE	MRE	RMSE
1	443	111	0.01533	0.00024	0.02105
2	343	86	0.03731	0.00091	0.05030
3	124	31	0.02471	0.00049	0.02726
4	842	211	0.03820	0.00043	0.07692
5	378	95	0.01738	0.00035	0.02260
6	205	52	0.07945	0.00182	0.09917
7	214	54	0.03236	0.00042	0.04339
8	957	240	0.06723	0.00099	0.08341
9	162	41	0.02549	0.00041	0.03305
10	1716	429	0.01151	0.00020	0.01589
11	1716	429	0.01566	0.00027	0.02129
12	1949	488	0.01630	0.00032	0.02227
13	479	120	0.05181	0.00143	0.06783
14	722	181	0.02498	0.00055	0.03801
15	1566	392	0.01719	0.00036	0.02584
16	381	96	0.01287	0.00022	0.01781
17	139	35	0.03270	0.00041	0.04183
18	1949	488	0.01059	0.00018	0.01590
AVG ± STD	\	\	0.02950 ± 0.01948	0.00056 ± 0.00045	0.04021 ± 0.02548

Table 6
Results of repeated experiments under different values of α.

Average metric	α					
	0	0.2	0.4	0.6	0.8	1
MAE	3.57754	0.03936	0.0237	0.02480	0.01853	0.08846
MRE	0.06453	0.00075	0.00045	0.00048	0.00036	0.00165
RMSE	4.58084	0.05046	0.03329	0.03309	0.02651	0.09980

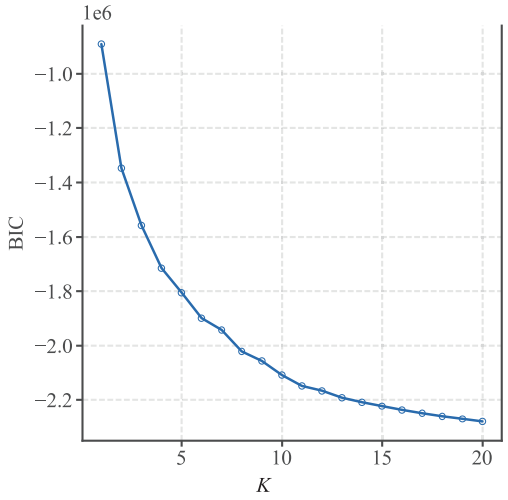

Fig. 7. Relationship between K and BIC in GMM clustering in the first experiment.

reliability of the source data matching process.

Although the TSAN identified 7 clusters in the first experiment, several clusters correspond to similar degradation behaviors from an engineering perspective. Therefore, to interpret the trend groups from an engineering perspective, three representative groups with clear correspondence are selected for analysis. Fig. 8 displays 30 representative raw EGTM trajectories selected from the identified clusters. The distinct EGTM ranges and temporal patterns show that the model effectively separates the data according to different stages of engine performance deterioration:

a) *Early-Life Phase* (Fig. 8(a)) This group maintains a high EGTM level (approximately 90–110 °C), indicating that the engine is in a mildly degraded condition. In this stage, compressor and turbine

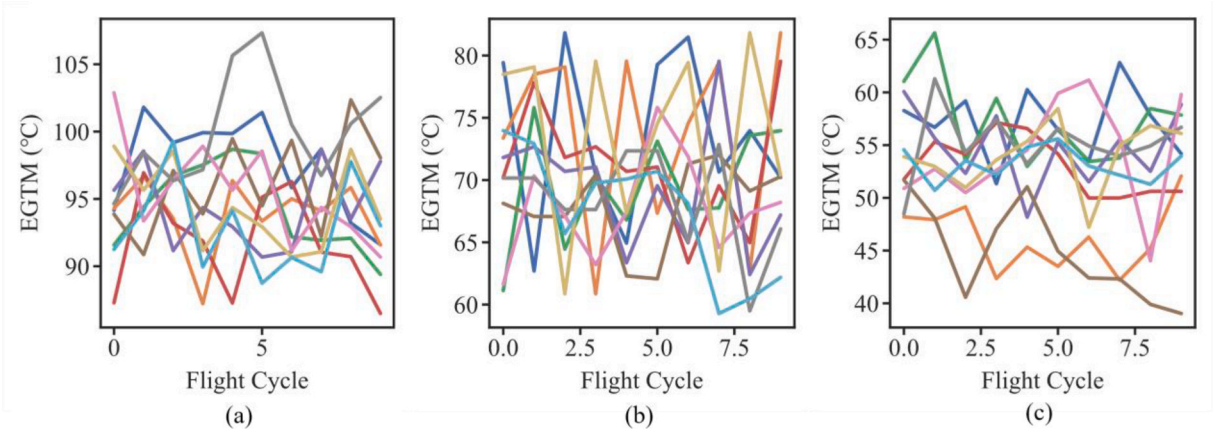

Fig. 8. Visualization of raw EGTM trajectories for the three identified trend groups. The distinct EGTM levels and temporal patterns reflect different stages of engine degradation: (a) early-life, (b) mid-life, and (c) late-life.

components preserve high aerodynamic and thermal efficiency. As a result, the EGT remains well below its operational limit, producing a sufficiently large temperature margin. The trajectories fluctuate mildly but consistently stay in the high-EGTM region.

b) *Mid-Life Phase* (Fig. 8(b)) The EGTM values narrow to a moderate range (60–80 °C), suggesting that noticeable degradation has started to accumulate. Typical engineering mechanisms include early-stage compressor blade erosion, minor fouling, and increased tip clearance, all of which reduce the isentropic efficiency. These effects elevate the EGT and reduce the available temperature margin. The trajectories exhibit moderate variability, consistent with engines undergoing progressive deterioration.

c) *Late-Life Phase* (Fig. 8(c)) In this cluster, the EGTM drops significantly (below 60 °C), reflecting near end-of-life conditions. Substantial compressor fouling, turbine efficiency losses, or flow-path deterioration increase the thermal load, pushing the EGT closer to its limit. To maintain the required fan speed, the control system typically compensates with higher fuel flow, further elevating EGT and causing the EGTM to remain at the lowest level among the groups.

Table 7 presents the experimental results of TSAN, which outperform those of the Separate GRU. The MAE, MRE, and RMSE decreased to 0.01853, 0.00036, and 0.02651, respectively. Compared with the best results of the baseline method across the three metrics, the proposed method achieved reductions of 36.0%, 35.7%, and 34.1% in MAE, MRE, and RMSE, respectively. Overall, TSAN reduces the standard deviations of MAE, MRE, and RMSE by 18.0%, 11.1%, and 15.2%, respectively, showing clearly improved stability over Separate GRU. From the perspective of MAE across different engines, TSAN performed slightly worse than the Single GRU only on Engine 8, with a 4.8% degradation, which is not significant. In terms of MRE, TSAN was inferior to the Single GRU only on Engine 8 by 2.3%, also insignificant. Regarding RMSE, TSAN showed a 4.8% decrease in performance compared to the Separate GRU on Engine 3, a 0.5% decrease compared to the Single GRU on Engine 8, and decreases of 4.9% and 8.0% compared to the Single GRU and Separate GRU, respectively, on Engine 13, all of which are relatively minor differences. It is worth noting that, over ten repeated experiments, the average number of fine-tuned models was 8.4, representing a 53% reduction compared with fine-tuning from the perspective of individual engines.

Table 7
Experimental results of TSAN.

No.	Total cycles	Testing cycles	MAE	MRE	RMSE
1	443	111	0.00829	0.00013	0.01107
2	343	86	0.02092	0.00052	0.03104
3	124	31	0.02040	0.00040	0.02856
4	842	211	0.01428	0.00016	0.03433
5	378	95	0.00856	0.00017	0.01127
6	205	52	0.06290	0.00141	0.08270
7	214	54	0.02199	0.00028	0.03122
8	957	240	0.03000	0.00044	0.03951
9	162	41	0.01237	0.00020	0.01573
10	1716	429	0.00543	0.00009	0.00721
11	1716	429	0.00597	0.00010	0.00785
12	1949	488	0.00869	0.00017	0.01195
13	479	120	0.05174	0.00140	0.07329
14	722	181	0.01377	0.00030	0.02262
15	1566	392	0.01163	0.00024	0.01906
16	381	96	0.00605	0.00010	0.00774
17	139	35	0.02494	0.00031	0.03363
18	1949	488	0.00567	0.00009	0.00845
AVG ± STD	\	\	0.01853 ± 0.01598	0.00036 ± 0.00040	0.02651 ± 0.02161

Since the TSAN utilizes unsupervised learning to generate trend sets adaptively based on data distributions, there are no fixed, pre-defined class labels across different experiments. To verify the validity of these dynamically generated clusters and the necessity of the separate fine-tuning strategy, a comparative "mismatch" experiment was conducted. In this experiment, instead of assigning the test samples to their correctly identified trend set models, a prediction network was randomly selected from the pool of fine-tuned models. The experimental results yielded an MAE of 0.05223, MRE of 0.00094, and RMSE of 0.06667. These errors remain markedly higher than the levels attained by the TSAN as shown in Table 6.

3.5. Comparative experiments

In addition to the baseline results discussed earlier, this study employed Transformer [31], Informer [32], and Multivariate Time Series Forecasting with Graph Neural Networks (MTGNN) [33] as comparative methods to evaluate the performance of TSAN. Furthermore, to verify whether its effectiveness stems from the core assumptions proposed in this paper rather than the advantages of the network architecture itself, an additional comparator, TSAN-NC, was constructed. This method shares the same network architecture as TSAN but does not distinguish sample groups during training and does not perform fine-tuning. For stability of the findings, each of the methods was executed 10 times, and the prediction outputs were averaged.

3.5.1. Comparison of prediction metrics

Table 8 summarizes the average results of the proposed and baseline methods across the three evaluation metrics. As illustrated, TSAN attains the best overall performance, whereas Informer yields the weakest

Table 8
Comparison of experimental results of 7 methods.

Average metric	Method						
	Single GRU	Separate GRU	TSAN-NC	TSAN	Transformer	Informer	MTGNN
MAE	0.02895	0.02950	0.02597	0.01853	0.19199	0.40826	0.09393
MRE	0.00057	0.00056	0.00052	0.00036	0.00347	0.00797	0.00181
RMSE	0.04023	0.04021	0.03462	0.02651	0.25443	0.55804	0.12728

results. It is worth noting that TSAN outperforms TSAN-NC, achieving reductions of 28.6%, 30.8%, and 23.4% in MAE, MRE, and RMSE, respectively. This further validates that the superiority of TSAN primarily originates from its underlying assumptions rather than its

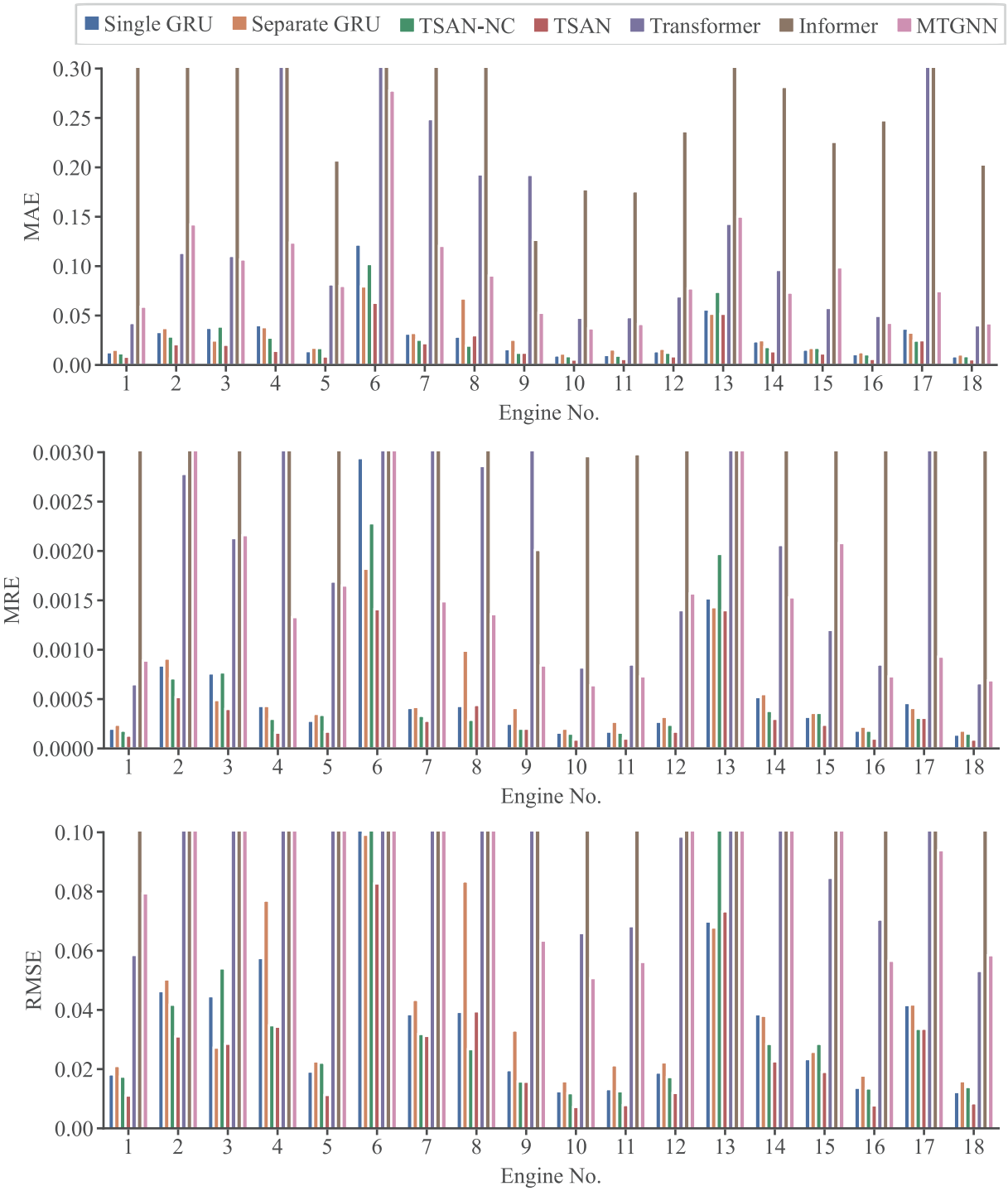

Fig. 9. MAE, MRE, and RMSE of prediction errors for each engine.

network structure alone.

Fig. 9 compares the predictive performance of the 7 methods across different engines under 3 evaluation metrics. Transformer and Informer models perform poorly on most engines, whereas MTGNN shows relatively better results. To further quantify the improvement of TSAN, particularly for engines with limited data, the results on 6 engines with the smallest data volumes among the 18 engines were selected and evaluated on three metrics. The percentage reductions in loss relative to Single GRU were computed and then averaged. The results show that for engines No. 3, 17, 9, 6, 7, and 2, the engine-level fine-tuning approach reduced the loss on MAE by 0.98%, while TSAN achieved a 36.05% reduction, representing an improvement of 35.07% over the fine-tuning baseline. For MRE, the baseline method improved by 1.62%, whereas TSAN achieved a 36.97% reduction, outperforming the baseline by 35.35%. In terms of RMSE, the baseline method reduced the loss by 0.22%, while TSAN achieved a 30.86% improvement, corresponding to a 31.08% enhancement over the engine-level fine-tuning method.

The results of each engine on the three metrics were normalized to evaluate dispersion and potential outliers. They were then visualized with box plots in Fig. 10. To facilitate comparison among different methods, only the range of 0 to 0.2 within the normalized interval of 0 to 1 is displayed. In terms of the MAE metric, TSAN achieves the lowest median value along with a shorter interquartile range and whisker length, indicating not only superior overall prediction accuracy but also greater stability across individual engines. Single GRU, Separate GRU, and TSAN-NC show similar yet relatively inferior performance on this metric, while Transformer, Informer, and MTGNN appear less suitable for this task. Although all methods exhibit certain outliers, TSAN shows the smallest magnitude of outliers, further showing its robustness. For the MRE and RMSE metrics, TSAN also achieves consistently favorable results. Overall, these findings indicate that TSAN outperforms the other methods in both prediction accuracy and stability across individual engines.

Fig. 11 shows the prediction results on the test set for the engines with the smallest and largest amounts of data, obtained using the seven methods described above. The differences between the 10-run average predictions and the ground truth are small relative to the fluctuations in the original values. Therefore, these differences are amplified by a factor of 80 to enhance visual comparability. The transformation is defined as:

$$\widetilde{\boldsymbol{y}}_s = \boldsymbol{y} + 80 \times \left(\widetilde{\boldsymbol{y}}^{avg} - \boldsymbol{y}\right) \tag{20}$$

where $\widetilde{\boldsymbol{y}}_s$ denotes the prediction results plotted in Fig. 11, and $\widetilde{\boldsymbol{y}}^{avg}$ is the mean prediction over 10 repeated runs for each method. In addition, since the prediction values of the Informer model exhibit excessively large deviations in some cases, the range of the plotted results has been truncated. As a result, the prediction points of certain methods may not appear within the displayed region of the figure.

Fig. 11(a) presents the experimental results for Engine 3, which has a relatively limited amount of data. In this figure, Informer and MTGNN exhibit predictions that deviate significantly from the ground truth. Although the Transformer shows smaller deviations overall, its predictions display consistent minor biases across most points. The Single GRU and TSAN-NC perform well on the majority of points, though both exhibit large deviations at a few specific instances. The Separate GRU, in contrast, shows no large deviations but maintains small systematic biases throughout. These observations indicate that engine-level fine-tuning can help mitigate large prediction errors, although its overall accuracy remains inferior to that of TSAN.

Fig. 11(b) presents the prediction results for Engine 12, which has a relatively large amount of data. Due to the high number of cycles in the test set, making it difficult to visualize the overall data distribution, only the first 30 flights are shown at the bottom for clearer comparison. As shown in the figure, the overall performance of Informer remains poor, while MTGNN and Transformer show noticeable improvements. It can be observed that, in most cases, the deviations among the first four methods are difficult to distinguish. For engines with larger sample sizes, the differences in prediction deviations are less pronounced than those observed for engines with smaller sample sizes.

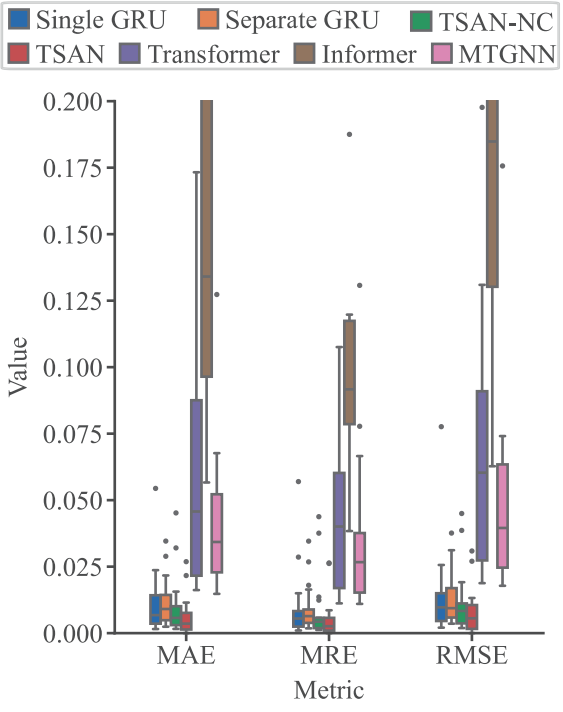

Fig. 10. Box chart of 7 methods under different metrics.

3.5.2. *Computational cost and analysis of system usability*

To comprehensively evaluate the resource consumption of different methods, the experiments were conducted on an Ubuntu 22.04 operating system with the following hardware configuration: an Intel(R) Core (TM) i7-9700 K CPU (8 cores, 8 threads), an NVIDIA RTX 4060Ti GPU (16 GB VRAM), and 16 GB of memory. The software environment consisted of Python 3.10 and PyTorch 2.1.1. Table 9 presents the training and testing durations of each method, where the testing duration represents the average result over 10 repeated runs. As shown in Table 9, TSAN requires a relatively longer inference time, and its training time does not exhibit a significant advantage.

Nevertheless, the TSAN is not intended for onboard deployment but rather for implementation within an airline's computing center. This design choice is based on the following considerations: (1) The predictive models for key performance parameters rely on data from multiple engines, which can be conveniently accessed and integrated within a data center environment. (2) The model training process can utilize the computing center's idle resources or be scheduled during off-peak periods through automated tasks, thereby avoiding interference with aircraft maintenance operations between landing and takeoff. (3) Furthermore, since multi-step degradation trajectories are obtained through recursive single-step prediction, performing such iterative predictions at the computing center can further ensure stable computing resources and improve consistency over an extended prediction range. Therefore, in practical applications, training time is not regarded as a critical factor. Although TSAN requires relatively more time for inference, its testing duration remains acceptable compared with the turnaround time between flight landing and subsequent departure.

4. Conclusion

Gas path performance parameters reflect the health condition of an engine. Therefore, accurately predicting their degradation trends is

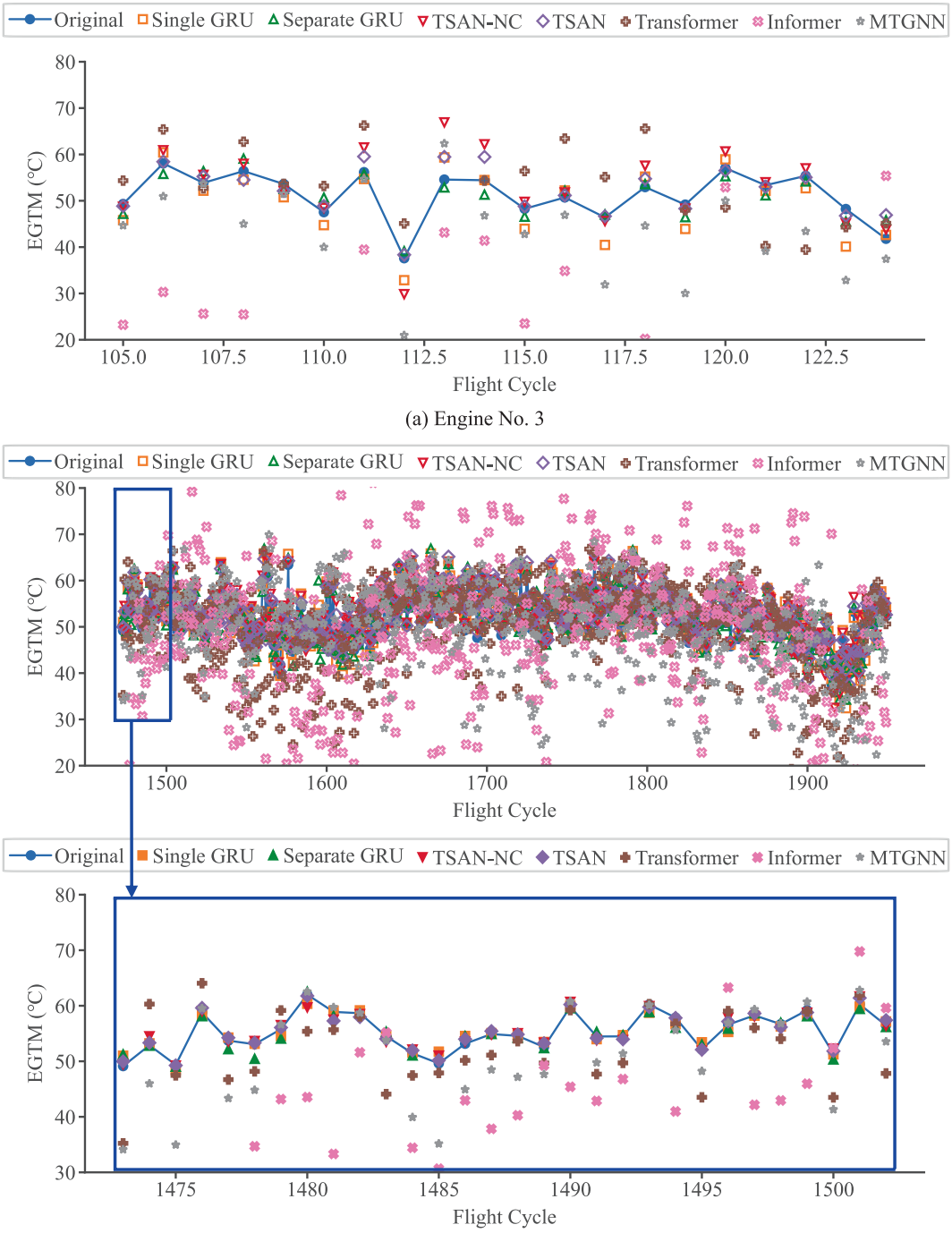

Fig. 11. Predictive performance of 7 models on 2 typical engines.

Table 9
Time consumption of different methods.

Method	Training time(s)	Testing time (s)
Single GRU	112.8	14.8
Separate GRU	1000.3	12.7
TSAN-NC	132.3	16.0
TSAN	238.2	51.3
Transformer	434.4	14.9
Informer	85.4	23.1
MTGNN	283.4	24.7

important for maintenance decision-making. However, current engine-specific modeling faces two major limitations: it requires building and maintaining a large number of individual models, and it often performs poorly on new engines due to limited historical data. To address these challenges, this paper proposes a Transferable Snippet Augmentation Network (TSAN), aimed at solving the prediction problem within a specific type of engine. By introducing a degradation trend representation module, the proposed method fine-tunes a unified base model from the perspective of degradation trends, thereby providing a scalable and practically deployable alternative to conventional engine-specific modeling schemes.

From an engineering implementation perspective, the proposed

method reduces the modeling and maintenance burden. Instead of maintaining a separate model for each engine, TSAN fine-tunes models based on a limited number of degradation-trend groups. In the experiments, the average number of fine-tuned models was 8.4, representing a 53% reduction compared with engine-specific fine-tuning. This reduction indicates that the proposed method enables a more efficient deployment strategy in industrial predicting systems, especially when a large number of engines must be monitored simultaneously.

In terms of predictive performance and robustness, TSAN consistently outperforms existing predicting methods under the experimental settings in this study. Compared with the Single GRU, Separate GRU, Transformer, Informer, and MTGNN models, TSAN improves MAE, MRE, and RMSE by 36.0%, 35.7%, and 34.1%, respectively. While engine-specific fine-tuning improves stability to some extent, TSAN achieves greater robustness, reducing the standard deviations of MAE, MRE, and RMSE by 18.0%, 11.1%, and 15.2%, respectively. Moreover, engines with scarce historical records tend to exhibit larger performance differences across predicting methods, while engines with more abundant data show relatively less variation. This observation indicates that small-sample engines may provide greater discriminative power and that TSAN appears to perform favorably in such data-limited scenarios in this study.

Further analysis in this study reveals that the performance improvement is primarily attributable to the accurate identification of degradation trend groups rather than increased model complexity. Results from the TSAN-NC ablation show that removing the trend representation module leads to a significant degradation in performance. Beyond the structural ablation, an implicit ablation was conducted by setting the weights of individual loss components to 0 or 1. The resulting performance degradation verifies that every module plays a necessary and irreplaceable role in the overall architecture. In addition, by comparing the two transfer perspectives, the results suggest that, in industrial equipment predicting, the degradation status of the equipment may have a more pronounced impact on prediction effectiveness than its individual variability.

Based on the above findings, while the proposed TSAN show improved accuracy, robustness, and deployment efficiency in data-limited scenarios, several issues revealed by the experimental results warrant further investigation. First, integrating mechanistic models to refine the physical interpretation of different trend groups may enable a more informed reassessment of the data partitioning strategy and clarify the origin of category-specific differences in real operating conditions. Second, the consistently inferior performance on engines 6 and 13 across multiple methods suggests potential factors such as sensor noise, abnormal readings, limited data coverage, or distributional shifts. As more detailed sensor-level information becomes available in future datasets, these hypotheses can be examined more thoroughly to identify the factors that truly drive the observed performance degradation. Third, while the current evaluation is limited to intra-type generalization, the proposed TSAN has the potential to extend to cross-type scenarios, as its degradation trend representations capture underlying physical degradation mechanisms shared across different aero-engine types. However, cross-type prediction remains challenging due to heterogeneous sensor configurations that cause semantic mismatches in inputs, as well as differences in engine design and materials that lead to distinct degradation trajectories and distribution shifts. These limitations warrant further investigation in future studies.

CRediT authorship contribution statement

Haoze Wu: Writing – review & editing, Writing – original draft, Software. **Shisheng Zhong:** Funding acquisition. **Minghang Zhao:** Writing – review & editing, Supervision, Conceptualization. **Yongjian Zhang:** Data curation. **Xuyun Fu:** Data curation. **Song Fu:** Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Key R&D Program of China (2023YFB4302400), National Natural Science Foundation of China (92360308).

Data availability

Data will be made available on request.

References

[1] D. Huang, D. Zhou, X. Wei, H. Wang, X. Zhao, Gas path deterioration observation based on stochastic dynamics for reliability assessment of aeroengines, Reliab. Eng. Syst. Saf. 238 (2023) 109458, https://doi.org/10.1016/j.ress.2023.109458.
[2] S. Zhong, Z. Li, L. Lin, Y. Zhang, Aero-engine exhaust gas temperature prognostic model based on gated recurrent unit network, in, IEEE International Conference on Prognostics and Health Management (ICPHM) 2018 (2018) 1–5, https://doi.org/10.1109/ICPHM.2018.8448857.
[3] J. Jung, C. Son, A. Rimell, R.J. Clarkson, A framework for an ML-based predictive turbofan engine health model, Aerospace 12 (2025) 725, https://doi.org/10.3390/aerospace12080725.
[4] E.T. Turgut, Seasonal and segmental variations and peaks in aircraft engine exhaust gas temperature, Int. J. Aeronaut. Space Sci. (2025), https://doi.org/10.1007/s42405-025-01023-4.
[5] D. Xiao, S. Song, H. Xiao, Z. Wang, Predicting the performance status of aero-engines using a spatio-temporal decoupled digital twin modeling method, Adv. Eng. Inf. 65 (2025) 103218, https://doi.org/10.1016/j.aei.2025.103218.
[6] Y. Wang, X. Wang, Z. Wang, B. Zhao, J. Xu, Y. Zhao, A novel method for aero-engine map calibration using adaptation factor surface, Measurement 239 (2025) 115394, https://doi.org/10.1016/j.measurement.2024.115394.
[7] Y.G. Li, M.F.A. Ghafir, L. Wang, R. Singh, K. Huang, X. Feng, Nonlinear multiple points gas turbine off-design performance adaptation using a genetic algorithm, J. Eng. Gas Turbines Power 133 (2011) 071701, https://doi.org/10.1115/1.4002620.
[8] W. Zhou, S. Lu, J. Huang, M. Pan, Z. Chen, A novel data-driven-based component map generation method for transient aero-engine performance adaptation, Aerospace 9 (2022) 442, https://doi.org/10.3390/aerospace9080442.
[9] H.-J. Jin, Y.-P. Zhao, M.-N. Pan, A novel method for aero-engine time-series forecasting based on multi-resolution transformer, Expert Syst. Appl. 255 (2024) 124597, https://doi.org/10.1016/j.eswa.2024.124597.
[10] Z.A. Shabbir, R.F. Swati, N. Ahmad, F.T. Zehra, S.R. Qureshi, A.A. Khan, Degradation classification of turbomachinery in high bypass ratio turbofan engine using supervised learning algorithms, J. Braz. Soc. Mech. Sci. Eng. 47 (2025) 152, https://doi.org/10.1007/s40430-025-05468-2.
[11] A. Kumar, A. Srivastava, N. Goel, J. McMaster, Exhaust gas temperature data prediction by autoregressive models, in: In: 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), 2015, pp. 976–981, https://doi.org/10.1109/CCECE.2015.7129408.
[12] D. Simon, A comparison of filtering approaches for aircraft engine health estimation, Aerosp. Sci. Technol. 12 (2008) 276–284, https://doi.org/10.1016/j.ast.2007.06.002.
[13] L.-H. Ren, Z.-F. Ye, Y.-P. Zhao, A modeling method for aero-engine by combining stochastic gradient descent with support vector regression, Aerosp. Sci. Technol. 99 (2020) 105775, https://doi.org/10.1016/j.ast.2020.105775.
[14] V. Zaccaria, M. Stenfelt, I. Aslanidou, K.G. Kyprianidis, Fleet monitoring and diagnostics framework based on digital twin of aero-engines, in, in: Proceedings of the ASME Turbo Expo 2018: Turbomachinery Technical Conference and Exposition, 2018, https://doi.org/10.1115/GT2018-76414.
[15] D. Xiao, Z. Lin, A. Yu, K. Tang, H. Xiao, Data-driven method embedded physical knowledge for entire lifecycle degradation monitoring in aircraft engines, Reliab. Eng. Syst. Saf. 247 (2024) 110100, https://doi.org/10.1016/j.ress.2024.110100.
[16] L. Lin, W. He, S. Fu, C. Tong, L. Zu, Novel aeroengine fault diagnosis method based on feature amplification, Eng. Appl. Artif. Intel. 122 (2023) 106093, https://doi.org/10.1016/j.engappai.2023.106093.
[17] M. Ilbas, M. Turkmen, Estimation of exhaust gas temperature using artificial neural network in turbofan engines, J. Therm. Sci. Technol. 32 (2012) 11–18.
[18] O.O. Dursun, S. Toraman, H. Aygun, Deep learning approach for prediction of exergy and emission parameters of commercial high by-pass turbofan engines, Environ. Sci. Pollut. Res. 30 (2023) 27539–27559, https://doi.org/10.1007/s11356-022-24109-y.
[19] H. Aygun, O.O. Dursun, K. Dönmez, O. Sahin, S. Toraman, Prediction of performance characteristics of an experimental micro turbojet engine using machine learning approaches, Energy 313 (2024) 133997, https://doi.org/10.1016/j.energy.2024.133997.

[20] D. Xiao, H. Xiao, Application of CNN-IndRNN model combined with residual structure in aircraft engine EGT prediction, J. Phys. Conf. Ser. 2764 (2024) 012009, https://doi.org/10.1088/1742-6596/2764/1/012009.
[21] X. Yu, Z. Zhang, B. Tang, J. Ma, Non-temporal neural networks for predicting degradation trends of key wind-turbine gearbox components, Renew. Energy 243 (2025) 122438, https://doi.org/10.1016/j.renene.2025.122438.
[22] L. Lin, J. Liu, H. Guo, Y. Lv, C. Tong, Sample adaptive aero-engine gas-path performance prognostic model modeling method, Knowl.-Based Syst. 224 (2021) 107072, https://doi.org/10.1016/j.knosys.2021.107072.
[23] Z. Yan, S. Zhong, L. Lin, Z. Cui, M. Zhao, A step parameters prediction model based on transfer process neural network for exhaust gas temperature estimation after washing aero-engines, Chin. J. Aeronaut. 35 (2022) 98–111, https://doi.org/10.1016/j.cja.2021.07.035.
[24] D. He, E. Bechhoefer, A. Hess, Automated rotorcraft turboshaft engine performance prediction using a transfer learning approach, in, IEEE Aerospace Conference 2025 (2025) 1–9, https://doi.org/10.1109/AERO63441.2025.11068693.
[25] Y. Xiao, H. Shao, S. Yan, J. Wang, Y. Peng, B. Liu, Domain generalization for rotating machinery fault diagnosis: a survey, Adv. Eng. Inf. 64 (2025) 103063, https://doi.org/10.1016/j.aei.2024.103063.
[26] X. Liu, L. Liu, L. Wang, X. Peng, Improved condition monitoring of aircraft auxiliary power unit based on transfer learning, in: In: 2019 Prognostics and System Health Management Conference (PHM-Qingdao), 2019, pp. 1–6, https://doi.org/10.1109/PHM-Qingdao46334.2019.8942809.
[27] M. Yan, N. Jiang, N. Li, TML-DA: transfer metric learning based distribution alignment framework for domain class imbalanced classification, Knowl.-Based Syst. 311 (2025) 113086, https://doi.org/10.1016/j.knosys.2025.113086.
[28] J. Chen, H. Jing, Y. Chang, Q. Liu, Gated recurrent unit based recurrent neural network for remaining useful life prediction of nonlinear deterioration process, Reliab. Eng. Syst. Saf. 185 (2019) 372–382, https://doi.org/10.1016/j.ress.2019.01.006.
[29] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, (2015). https://doi.org/10.48550/arXiv.1503.02531.
[30] X. Zheng, H. Shao, S. Yan, Y. Xiao, B. Liu, Multiscale information enhanced spatial-temporal graph convolutional network for multivariate traffic flow forecasting via magnifying perceptual scope, Eng. Appl. Artif. Intel. 136 (2024) 109010, https://doi.org/10.1016/j.engappai.2024.109010.
[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2017. https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.
[32] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence 35, 2021, pp. 11106–11115, https://doi.org/10.1609/aaai.v35i12.17325.
[33] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, C. Zhang, Connecting the dots: multivariate time series forecasting with graph neural networks, (2020). https://doi.org/10.48550/arXiv.2005.11650.