Continual contrastive reinforcement learning: Towards stronger agent for environment-aware fault diagnosis of aero-engines through long-term optimization under highly imbalance scenarios

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

Abstract: Although the stability of aero-engines is high, their failures can lead to catastrophic consequences. Due to the infrequent nature of faults, traditional data-driven fault diagnosis methods rely on limited amounts of historical failure data for training classification models. They cannot update models on time in response to environmental changes and data growth. To address the issue, this paper proposes a new machine learning method, i.e., Continual Contrastive Reinforcement Learning (CCRL), that integrates environmental interaction and continual dynamic evolution for fault diagnosis of aero-engine under conditions of high imbalance and continually growing data. First, the operating environment of the airline is treated as the learning environment for the agent. The aircraft's flight data is used as the state information provided by the environment, while the failure identification results from ground personnel and experts serve as the labels for this state information. This framework ensures the agent can continually learn in the face of increasing data volumes. Next, a contrastive learning encoder for highly imbalanced scenarios is designed, where a large number of normal samples are used to train an encoder that constructs positive and negative sample pairs with actual data, fine-tuning the encoder to improve its ability to distinguish different faults. Finally, the contrastive learning encoder is embedded into the enhanced learning model, enabling the agent to better perceive environmental changes and diagnose failures under highly imbalanced scenarios. This paper conducts a series of contrastive and ablation experiments using real data, which fully validate the application potential of the proposed method.

Keywords: aero-engine fault diagnosis, continual contrastive reinforcement learning, environment awareness, monitoring data growth

1 Introduction

Aero-engines serve as the primary power source for aircraft. Due to the prolonged

* Corresponding authors.
E-mail addresses: 21b908077@stu.hit.edu.cn (H. Wu), zhongss@hit.edu.cn (S. Zhong), zhaomh@hit.edu.cn (M. Zhao), fuxuyun@hit.edu.cn (X. Fu) , zhangyj@hitwh.edu.cn (Y. Zhang), fusong@hit.edu.cn (S. Fu)

exposure to internal high temperatures and pressures, as well as external extreme climatic conditions and erosion by impurities, some components are prone to encounter failures such as blades, gas paths, lubrication systems, and sensors[1,2]. These faults lead to at least two major issues: First, they significantly compromise the reliability of aero-engines. If a failure occurs during high-altitude operations, it may pose hidden safety risks to the flight. Likewise, failure to investigate fault signs on time could result in catastrophic consequences and significant economic losses. Secondly, due to the complex structure of aero-engines, a single fault phenomenon may correspond to multiple fault types, leading to time-consuming diagnostic processes and reduced flight efficiency. Therefore, it is crucial to promptly assess potential failures based on early-stage indicators of engine malfunction, to identify and mitigate risks on time[3].

The fault diagnosis methods for aero-engines can be broadly categorized into diagnostic methods based on mechanistic models and data-driven approaches. Mechanistic-based methods establish normal state or typical faults simulation models for specific components or systems. Fault types can be identified by analyzing the differences between actual data and the simulation model [4]. Tan et al.[5] proposed an approach to evaluate the health status of multiple components in aero-engines. They established an algebraic system model for each component based on its status data and systematically organized these models. The health condition is estimated based on the trend of curve changes. Zhang et al. [6] employed a collaborative and sparse classification framework leveraging prior knowledge to diagnose gear hub cracks in aero-engines. However, there are still limitations in diagnosis based on mechanistic models. Firstly, the actual operating conditions of aero-engines often introduce complexities beyond what theoretical models can account for. Secondly, the nonlinear relationships between data and faults, as well as between different types of faults, pose challenges for fault diagnosis when relying on specific models. Thirdly, developing precise mechanism models requires researchers to possess extensive knowledge of aero-engines and access to comprehensive test data [7]. Therefore, the application of mechanism-based fault diagnosis methods currently faces several challenges[8].

In contrast, data-driven fault diagnosis methods do not rely solely on domain-specific knowledge of aero-engines. Instead, they utilize sensor data collected during operation and fault reports generated throughout the running process for fault diagnosis. The basic process of a data-driven fault diagnosis method typically involves three steps: first, collecting and sampling data from various health states to establish a known sample set; second, using the known sample set to train a machine learning model, which then classifies the input samples based on their fault types; finally, the trained fault diagnosis model is applied to actual fault diagnosis tasks [9]. Wen et al.[10] proposed a layered convolutional neural network, where a shared CNN is first used to extract fault features from the data, followed by two branch networks to diagnose both the fault mode and its severity. Liu et al.[11] analyzed the statistical process of aero-engine bearings, dividing them into multiple levels of signals. Then, an LSTM model is used to identify the fault types and their severity in the bearing system. Lee et al.[12] incorporated an attention mechanism into a recurrent neural network, using current and corresponding speed signals as inputs to diagnose faults in permanent magnet

synchronous motors. However, traditional machine learning methods typically assume that the numbers of training samples for different fault types are balanced. In contrast, aero-engine failures are rare, with a limited number of faulty samples and a wide variety of failure types. If an under-sampling method is applied to reduce the number of majority class samples to match the number of samples from minority fault types, the overall sample size may become too small, leading to poor model performance[13].

To address the challenges of insufficient samples and class imbalance, data augmentation and transfer learning are commonly adopted solutions. The core principle of data augmentation is to transform and generate additional data based on the original dataset. In image processing, significant diversity can be achieved while retaining semantic similarity through techniques such as flipping, rotating, and adjusting local colors and structures[14,15]. For time series data augmentation, methods such as Synthetic Minority Over-Sampling Technique (SMOTE), random sampling, generative adversarial networks (GANs), and self-supervised learning have been proposed[16]. Kamycki et al.[17] proposed a data augmentation method for distorted time series based on the dynamic time warping (DTW) algorithm, which enriched time series samples for classification. Inspired by SMOTE, Liu et al [18] proposed an extrapolation-based method in the feature space to generate additional data. Lu et al.[19] proposed a GAN-based time series data augmentation method leveraging Wasserstein distance metrics. On the one hand, GAN-based methods often face convergence issues. On the other hand, unlike image-oriented data augmentation, there is still no solid theoretical evidence that these transformations applied to time series data can correctly and reliably capture the universal features of samples[20].

Transfer learning can leverage a large number of samples from related fields to improve the generalization of models in specific tasks, while also mitigating the risk of overfitting due to limited samples [21]. The fundamental principle of transfer learning is that there are shared patterns between the source domain and the target domain. The model trained on the source domain is transferred to the target domain to reduce the need for a large number of samples in the target domain. This method is suitable for scenarios where there are few samples in the target domain, abundant samples in the source domain, and a high degree of similarity between the two domains. In recent years, several methods based on transfer learning have been proposed to address the issue of diagnosing faults with limited sample sizes. For example, Zhong et al.[22] proposed a transfer learning approach combining support vector machines and convolutional neural networks, where features are learned from a large number of engine samples and transferred to a smaller set of samples for fault diagnosis. Chen et al.[23] applied two transfer learning algorithms to diagnose two types of faults in fan sensor data collected from the field. However, transfer learning requires a large number of reference samples, while failure samples for aero-engines are limited for most engine types. Additionally, transfer learning assumes that samples from the source domain can be effectively applied to the target domain, but measuring the similarity between the source and target domains is often challenging. When introducing a large number of samples to augment a small sample set, there is a risk that the model may learn incorrect prior knowledge. Aero-engine failures have a low incidence and diverse fault types,

leading to an imbalance, with more normal samples than faulty samples. The limited number of faulty samples contains a wide variety of fault types, making it a highly imbalanced multi-class classification problem [24]. As discussed above, from the perspective of data augmentation, generating more new data can be challenging, as it is difficult to ensure that the newly generated time series follow the same patterns as the existing data, while preserving the distinct characteristics of different fault types. Additionally, when transferring data from different types of engines, it is challenging to ensure that the model learns the common characteristics between the engines, rather than acquiring erroneous prior knowledge. Furthermore, in real-world scenarios, airlines operate a large number of aircraft performing flight missions daily, leading to continually growing data amount. Currently, most traditional neural network training and deployment processes are relatively independent, and models cannot be continually updated to adapt to changing environments, which brings inconvenience to actual application.

Without generating new data or relying on the characteristics of other engine types, reinforcement learning can address the issue of sample imbalance in aero-engine fault diagnosis. In reinforcement learning-based fault diagnosis, the problem is transformed into a fault prediction scenario. Fault sample information from sensors is input into the agent, which learns by predicting fault types and receiving feedback from its predictions. Martinez et al.[25] enhanced the importance of agents to temporal data by assigning different weights to early and recent data. Ding et al.[26] constructed a fault prediction game scenario and applied deep Q-learning for fault diagnosis. Lin et al.[27] used an unbalanced reward mechanism to halt training immediately when errors occurred in small-sample training, thereby mitigating the bias of the agent across different sample types. Therefore, reinforcement learning can be designed to adjust the weights of different fault types, enabling comprehensive utilization of diverse fault samples in aero-engine fault diagnosis.

Although reinforcement learning can address the problem of sample imbalance, the limited number of faulty samples in aero-engine fault diagnosis tasks still makes it challenging for reinforcement learning models to accurately distinguish different fault types. Contrastive learning can learn discriminative feature representations by maximizing the similarity of positive pairs while minimizing the similarity of negative pairs. As a result, this method improves sample efficiency, enabling effective distinction between different categories even with a limited number of samples. For example, Wang et al. [28] applied contrastive learning in medical image analysis to learn richer feature representations from a large number of unlabeled image samples, enhancing the utility of learned features in downstream tasks. Oquab et al. [29] employed unsupervised contrastive learning on unlabeled image data, enabling the model to learn general-purpose visual representations without the need for manual annotations. Chen et al.[30] proposed a simple framework for contrastive learning of visual representations (SimCLR), which constructs positive and negative samples using augmented views of the same image and different images, respectively. This approach enhances the aggregation of features for similar images while improving the separation of features for different image categories. However, the implementation of SimCLR

still relies on data augmentation techniques to generate additional samples. While this approach is reasonable in the field of image processing, it may encounter challenges when directly applied to time-series data.

To address this, this paper proposes a machine learning method with capabilities of environmental interaction and continual dynamic evolution, termed Continual Contrastive Reinforcement Learning. This method is specifically designed for real-world fault diagnosis tasks under conditions of dynamic operational environments, highly imbalanced fault classes, and continually growing data amount. To better capture the features, actual faulty samples of the same type are designated as positive samples, while those of different types are designated as negative samples to train the contrastive learning network. To address the scarcity of faulty samples, which makes it challenging to capture the patterns for the learning network, a large number of normal samples are used to pre-train the encoder part of the learning network. Subsequently, the encoder is fine-tuned during the contrastive learning process. To address the issue of sample imbalance and ensure the model can adapt to environmental changes in practical applications, the reward mechanism during agent training is adjusted based on the distribution of fault types in the training samples. A specifically designed integration of contrastive learning and deep reinforcement learning is employed to improve fault diagnosis performance. The main contributions of this paper are summarized as follows:

1) Using the operating environment of the airline as the learning environment for the agent, the developed method continually trains the agent for fault diagnosis under the condition of increasing data amount. It aims to strengthen the fault diagnosis ability, maintain the agent's adaptability to environmental changes, and enhance its ability to handle highly imbalanced samples.
2) A weighted contrastive learning loss function with hyperparameter adjustment is constructed, in which existing samples are used to construct positive and negative sample pairs, avoiding additional uncertainty introduced by generating new time-series data. This function enhances the clustering of features from the same type of faulty samples, suppresses interference from different types of faulty samples, and ultimately improves the model's stability in distinguishing fault types.
3) Before contrastive learning training, normal samples are used to pre-train the encoder part of the network. Subsequently, a fine-tuning method is applied during the contrastive learning process to mitigate the impact caused by the scarcity of fault samples.
4) A real aero-engine dataset is used to conduct contrastive and ablation experiments. The results show that the proposed method achieves good performance, and the effects of each module align with design expectations.

The remaining sections of this paper are organized as follows. Section 2 introduces the experimental dataset, and the pre-processing steps conducted prior to the experiments. Section 3 presents the proposed method, covering its overall structure, input and output design, detailed implementation, and the process of network training and testing. Section 4 describes the experimental design, testing platform, and discusses the results. The final section provides a summary of the paper.

2 Processing of experimental data

Fig. 1 illustrates the data collection process in fault diagnosis experiments. The left part of the figure represents the complete life cycles of C engines, from initial operation to disassembly due to faults. The green section of the timeline indicates the number of flights during which the engine operated without failure. The number of flights between the installation of each engine and the occurrence of a fault varies, resulting in different lengths of the green sections on the timeline. The right end of the green section in the timeline corresponds to the flight where a failure occurs, with different colors indicating different types of faults.

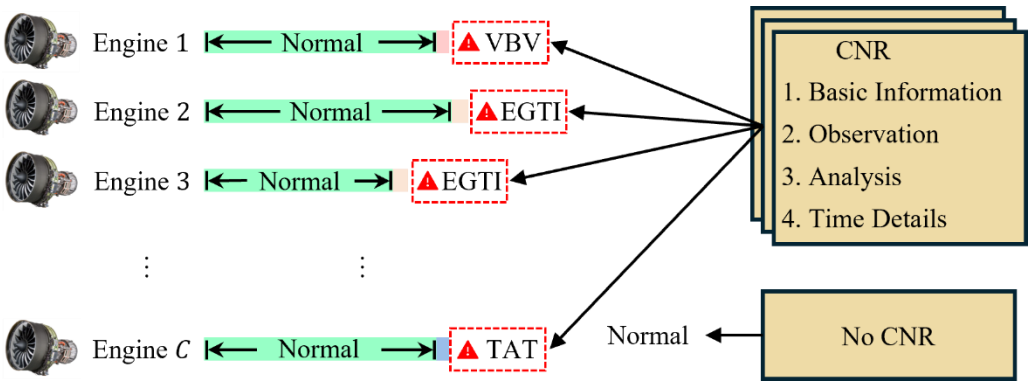

Fig. 1. Overview of the data collection process.

The right part of Fig. 1 represents the basis for determining whether a failure occurs, which is derived from the Customer Notification Report (CNR) provided by the original equipment manufacturer (OEM). The main page of the report contains four sections. The first section includes basic information such as customer names, engine models, aircraft types, serial numbers, installation positions, report time, case numbers, and priority levels. The second section highlights anomalies or numerical changes that deviate from predefined thresholds. The third section highlights instances where monitored values during engine operation exceed predefined limits or exhibit irregular patterns. This section builds upon these observations, leveraging an expert system to identify potential engine issues and assign fault type labels, which constitute the core focus of this paper. The final section provides time-related information, including engine installation time, operational duration, and the start and end times of data fluctuations.

To facilitate analysis, possible faults provided in the CNR report are labeled as fault types for the corresponding fault flights. Flights before receiving the CNR report are treated as normal flights. Since most faults occur within a specific number of flights, the flights observed at the end of the engine's operation can be considered as the fault stage. The key operational parameters and fault types provided in the CNR report are used to determine the fault of this stage. The remaining flights are considered normal. As shown in the red dotted frame in Fig. 1, this study focuses on three fault types: variable bleed valve (VBV) system failure, exhaust gas temperature imbalance (EGTI), and total air temperature (TAT) sensor failure. A dataset is constructed containing four

categories: normal samples, VBV fault samples, EGTI fault samples, and TAT fault samples. According to Ref. [31], four groups of performance parameters are collected for fault diagnosis tasks: Delta Exhaust Gas Temperature (ΔEGT), Delta Core Speed (ΔN_2), Delta Fuel Flow (ΔFF), and Fan Speed (N_1).

After obtaining the raw data, pre-processing steps must be completed, including dataset segmentation and data normalization. Once the dataset structure is defined, it is necessary to partition the data into training and test sets. In this study, an equal number of samples from each fault type are randomly selected as the test set, while the remaining samples are used for training. The training and test sets are then normalized to the range of $(0,1)$.

3 Methodology

In this section, the overall structure of the proposed method and its modules are introduced. First, the overall framework and the input of the network are presented. Subsequently, the network's feature distinction and type classification modules are described in detail. Finally, the processes of training and testing are explained.

3.1 Overall structure

Fig. 2 illustrates a schematic diagram of the overall CCRL process, highlighting the interaction between the agent and airline business processes in a real-world scenario. The left part of the figure represents the training process of CCRL, while the right part depicts a simplified structure that is continually updated. First, during business operations, new aero-engine sensor data is recorded for each aircraft after completing a flight. This data is transmitted to the ground via Aircraft Communications Addressing and Reporting System (ACARS) messages, a type of air-to-ground communication, and subsequently stored on disk. Samples for fault diagnosis consist of performance data collected at key stages of aero-engine operation. The performance data from the m-th flight can be represented as $\boldsymbol{x}_m = \{x_{\Delta EGT}, x_{\Delta N2}, x_{\Delta FF}, x_{N1}\}$. Second, based on recent data from M aero-engine flights, the performance data matrix is denoted as $\boldsymbol{X}_n^y = \{\boldsymbol{x}_1^y, \boldsymbol{x}_2^y, \dots, \boldsymbol{x}_M^y\}$, where n denotes the sample index. The agent evaluates whether the engine has failed after completing its task and identifies the type of failure. Third, experts rely on their experience and the outputs of the agent to determine the necessary maintenance actions. They verify whether the actual situation aligns with the system's judgment and evaluate the agent's performance based on predefined criteria. Here, the actual fault type is represented as y. Fourth, sensor data, judgments from the agent, and related work details are recorded in the experience library, along with the final evaluation and other relevant information. Finally, the agent learns from the experience library to enhance its accuracy in fault type identification. It is worth noting that the training and testing processes of the agent are not isolated. Once integrated into the workflow, the agent is dynamically updated as the airline's internal aero-engine data grows and diagnostic capabilities improve, enabling adaptation to new scenarios.

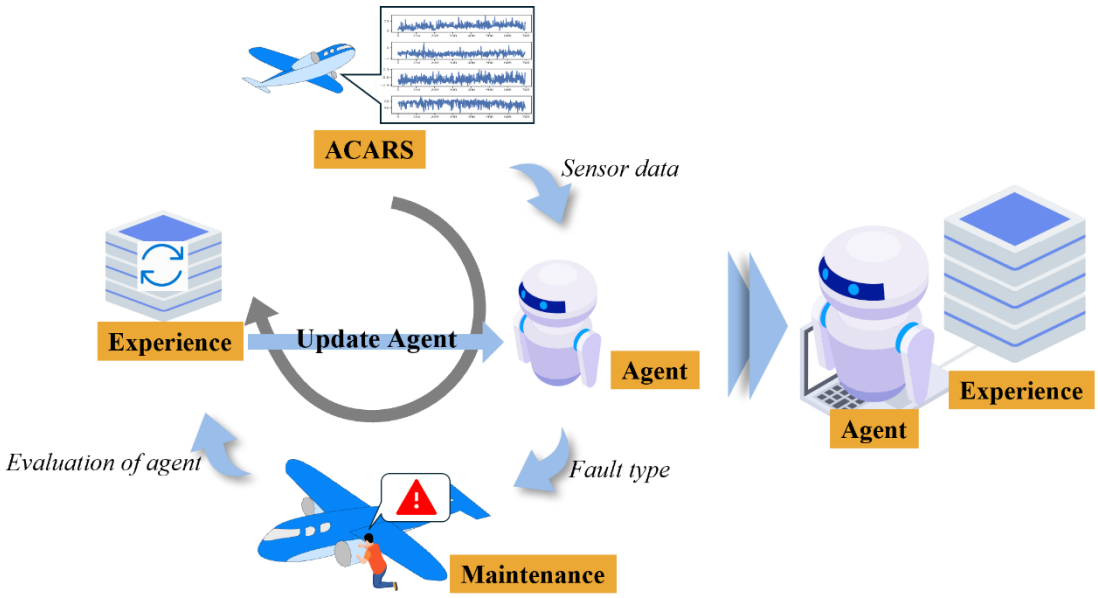

Fig. 2. Schematic diagram of the overall CCRL process.

3.2 Feature distinction module

Fig. 3 illustrates the schematic diagram of the feature distinction module. In general, for contrastive learning methods such as SimCLR, features obtained from image augmentation operations are used as positive and negative sample pairs. Positive samples refer to features derived from data augmentation at the same image, while negative samples are derived from data augmentation at different images. For time series data, although related research on data augmentation exists (e.g., TimeCLR [32]) , there is still no solid theoretical foundation ensuring the correctness of time-series data augmentation. Moreover, addressing the issue of insufficient labels is not essential for the scenario in this study. Therefore, different samples of the same failure type can be used as positive sample pairs. For samples of fault type $y = 1$, positive sample pairs can be expressed as $< \boldsymbol{X}_1^1, \boldsymbol{X}_2^1 >, < \boldsymbol{X}_1^1, \boldsymbol{X}_3^1 >, \dots, < \boldsymbol{X}_{N-1}^1, \boldsymbol{X}_N^1 >$. Negative sample pairs can be expressed as $< \boldsymbol{X}_1^1, \boldsymbol{X}_1^2 >, < \boldsymbol{X}_1^1, \boldsymbol{X}_2^2 >, \dots, < \boldsymbol{X}_{N-1}^1, \boldsymbol{X}_N^Y >$, where Y represents the total number of fault types. In contrastive learning, cosine similarity is used to evaluate the similarity of two samples in the feature space. This is defined as the dot product of two vectors divided by the second-order norm of the vectors, represented as $\mathrm{SIM}(\cdot,\cdot)$. The loss function of SimCLR is the normalized temperature-scaled cross entropy loss (NT-Xent). It is expressed as:

$$l_{i,j} = -log \frac{exp\left(\mathrm{SIM}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)}{\sum_{k=1}^{2N} 1_{[k \neq i]} exp(\mathrm{SIM}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)} \tag{1}$$

Here, $\boldsymbol{z}_i, \boldsymbol{z}_j$ represent the projections of the positive sample pair. τ is a parameter controlling the scaling of similarity. N denotes the size of the mini batch. $1_{[k \neq i]}$ equals 1 when $k \neq i$, and 0 otherwise. Since the sample pairs in this paper are constructed using real samples of different fault types, different fault types appear in each batch with a certain proportion. This results in an imbalance between positive and

negative sample pairs. Therefore, the loss function is adjusted as follows:

$$\mathcal{L} = -\frac{1}{P}\sum_{i=1}^{P} w_p \cdot log\frac{exp\left(\mathrm{SIM}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)}{exp\left(\mathrm{SIM}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right) + \sum_{k\neq i} exp\left(w_n \cdot \mathrm{SIM}\left(\boldsymbol{z}_i, \boldsymbol{z}_k\right)/\tau\right)} \quad (2)$$

Here, $\boldsymbol{z}_i, \boldsymbol{z}_j$ represent the projections of positive sample pairs. $\boldsymbol{z}_i, \boldsymbol{z}_k$ represent the projections of negative sample pairs. w_p is the weight for positive samples, w_n is the weight for negative samples, and P denotes the total number of positive samples. The reasons for setting weight parameters can be primarily attributed to two factors. First, in actual fault data, the number of samples for each fault type is relatively small, while the number of fault categories is large, leading to sample imbalance. Second, the NT-Xent loss function treats different samples as negative pairs. However, in time-series data, even samples from different fault types often follow the general patterns of engine parameters. Here, increasing positive sample weights enhances the clustering of feature projections for the same fault type. Conversely, reducing negative sample weights weakens the separation between feature projections of different fault types.

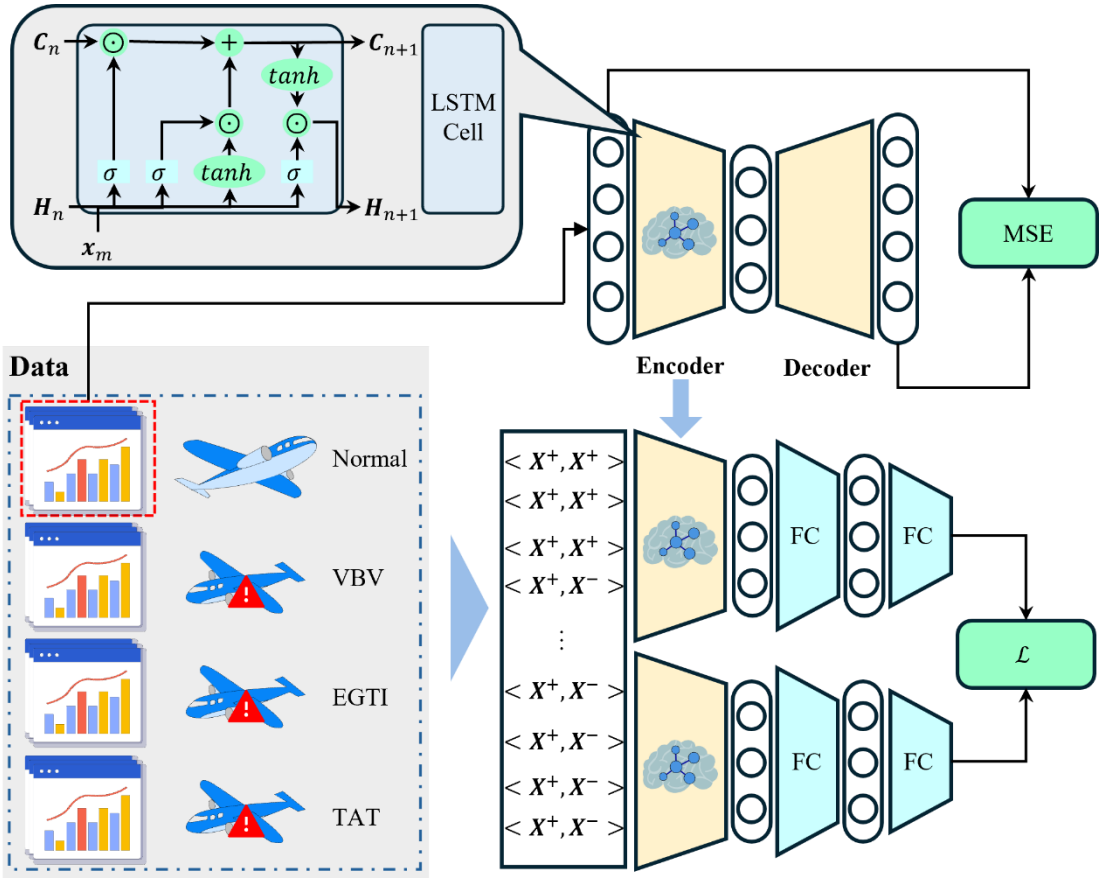

Fig. 3. Schematic diagram of the feature distinction module.

Since this method directly uses real samples to construct positive and negative sample pairs without data augmentation, it is necessary to address the issue of limited original training data. To address this, the encoder is pre-trained before contrastive learning to capture general patterns. First, a self-encoder is constructed, and only normal samples are used to train it, enhancing the encoder's data compression capability.

Only normal samples are used for training because aero-engines have high maintenance requirements, fault data is scarce and highly imbalanced, and each fault exhibits different characteristics. The autoencoder in this method is designed to better compress features, and selecting normal samples with the largest amount of data helps prevent the model from overfitting to specific fault types. Then, the trained encoder is used as the basis for contrastive learning. The contrastive learning network is trained by fine-tuning the encoder to reduce the projection distance of samples from the same fault type and increase the projection distance between samples of different fault types. In this study, LSTM is employed as the foundational network for contrastive learning. LSTM is a classic recurrent neural network. Its core principle is to sequentially process key data from each flight, using the hidden layer and current flight data to adjust the weights between current inputs and historical information. It also selectively forgets or updates historical information through its gating mechanisms. As LSTM is a well-established network, the implementation details are not discussed in this paper. In the self-encoder, as the entire time sequence information is required for contrastive learning, the final hidden state of the LSTM is used as the encoder's output and the decoder's input, enabling the encoder to learn fundamental patterns.

After training the self-encoder with normal samples, the encoder is employed as part of the contrastive learning network. A projection layer, composed of two fully connected layers, is added to form the contrastive learning network. During contrastive learning, a smaller learning rate is set for the encoder compared to the projection layer to avoid significant parameter adjustments during training. Fine-tuning allows the encoder to retain its ability to understand data characteristics while distinguishing different fault types.

3.3 Type identification module

The type identification module is primarily used to distinguish samples from different fault types. The specific training process is illustrated in Fig. 4(a). The primary goal is to adjust the weighting for different fault types and enhance the agent's ability to accurately identify fault type samples. This module is divided into three parts: the environment, the agent, and the learning process.

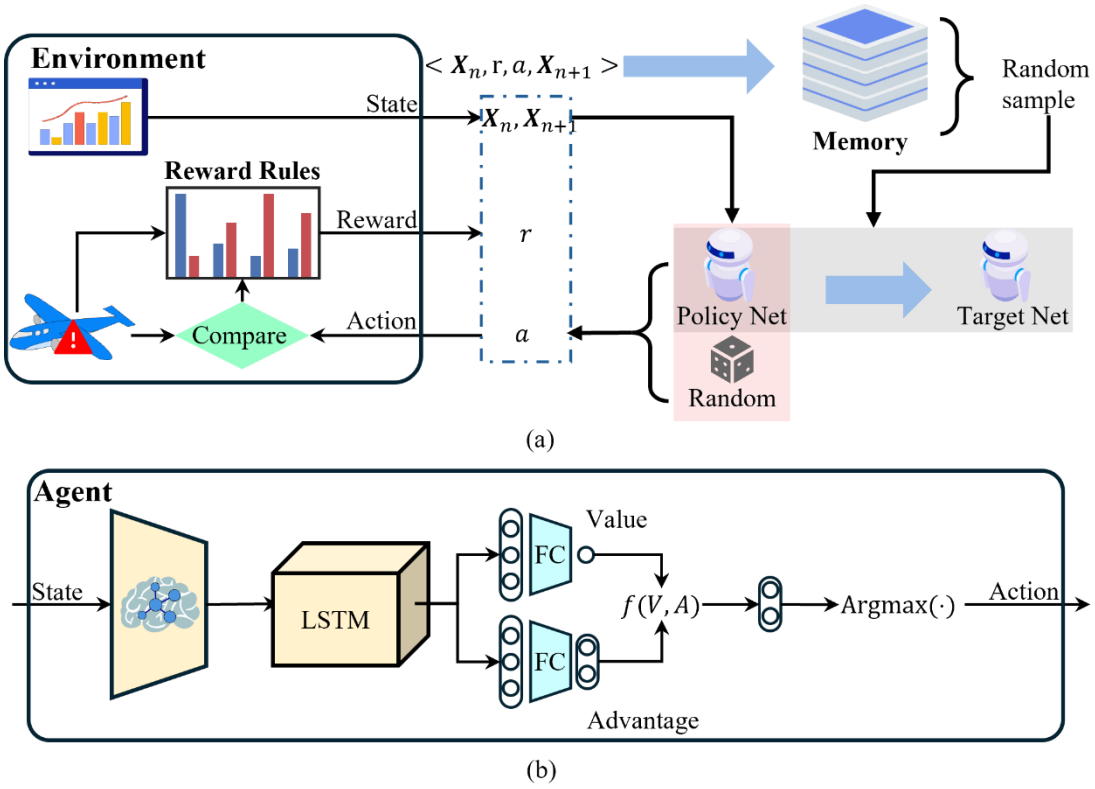

Fig. 4. Schematic diagram of type identification module: (a) Type identification module training process, (b) Policy net and target net structure in type identification module.

The type identification module employs the Dueling Double Deep Q-Network (D3QN). The basic principles of D3QN are as follows: first, the environment provides a state. Second, the agent acquires the state and estimates the value (i.e., the Q-value) for selecting a specific action in that state, which is then returned to the environment. Third, the environment processes the action, providing rewards and new states based on the action taken. Fourth, the agent processes the new state. This process is repeated iteratively, allowing the agent to continually optimize the Q-value estimation for each action in various states, thereby maximizing the overall cumulative reward.

3.3.1 Design of the environment

In this paper, the operational processes of the airline are treated as an "environment", which is assessed by the agent based on the provided samples. The basic workflow of the "environment" is illustrated below:

1) At the beginning of training, the environment needs to provide a set of samples (i.e., $\boldsymbol{X}_n$) to the agent. The environment internally retains the actual fault type y.
2) The agent determines the fault type of the samples (i.e., a). The environment receives the result returned by the agent.
3) The environment compares the agent's result with the internal true result y. Based on the correctness of the result, the environment provides a reward (i.e., r) according to the reward rules. If the next set of samples exists, the environment returns the next set of samples (i.e., $\boldsymbol{X}_{n+1}$).

4) Repeat steps 2 and 3 until all the data in an episode are processed.

For fault diagnosis involving imbalanced samples, the value of identification results for rare fault types cannot be treated the same as those for normal samples, necessitating specially designed reward rules. Specifically, this study adopts a formula like normalized weight allocation for reward calculation. Let $\{D_1, D_2, D_3, D_4\}$ represent the sample numbers for four fault types. The weight w_y for fault type y is defined as $1/D_y$. The absolute value of the reward is calculated as:

$$R_y = w_y / \sqrt{\sum_{y=1}^{Y} w_y^2} \tag{3}$$

The final reward r depends on whether the agent correctly identifies the fault type. It is expressed as:

$$r = \begin{cases} R_y & \text{if } y = a \\ -R_y & \text{if } y \neq a \end{cases} \tag{4}$$

3.3.2 Design of the agent

The structure of the agent is illustrated in Fig. 4(b). The input state to the environment corresponds to the samples collected by the sensor, as mentioned earlier. The output represents the fault type determined by the agent based on the collected samples.

The structure of the agent includes the encoder obtained from the steps described earlier, a new LSTM-based main network, and two linear layers. The encoder from the feature distinction module is no longer fine-tuned during the agent's training process. Instead, it is directly used to provide its output features as input to the new LSTM-based main network. Since the output of the new LSTM-based main network is used for classification, it corresponds to the output of the hidden layer at the final time step. To comprehensively evaluate the value of a state and avoid the overall impact of different fault types on action value estimation, two linear layers are used. One represents the value (i.e., V), indicating the overall value of taking any action in a given state. The other represents the advantage (i.e., A), indicating the superiority of selecting a specific action in that state. To combine the state value and the advantage of each action, the final Q-value is calculated using $f(V, A)$, expressed as:

$$f(V, A) = V + \left(A - \mathrm{AVG}(A)\right) \tag{5}$$

Here, V is a scalar, and A is a vector with a length equal to the number of fault types. $\mathrm{AVG}(\cdot)$ is used to calculate the average value of the vector. The output of $f(V, A)$ is a vector with a length equal to the number of fault types. The index of the maximum Q-value in the vector is selected. The fault type corresponding to this index is taken as the agent's classification result.

The structure in Fig. 4 (b) appears twice in Fig. 4 (a), namely as the Policy Net and the Target Net. These two networks, together with the exploration mechanism, constitute the decision-making strategy of the agent. The two networks serve different purposes in Q-value calculation: the Policy Net is primarily used to select actions based on the state, while the Target Net is used to compute the target Q-value. Since the Target Net is not updated in real time, the target values remain more stable, reducing

fluctuations during the learning process. The Policy Net, on the other hand, is updated in real time, ensuring that action selection during training is not entirely dictated by the Target Net's output. The collaborative computation mechanism of these two networks effectively mitigates the issue of Q-value overestimation in the intelligent agent.

3.3.3 Learning process of the agent

The learning process of the agent described in this paper is illustrated in the right part of Fig. 4(a) and mainly consists of two stages. The first stage focuses on fault type identification and sample recording. During this stage, the environment interacts with the Policy Network to generate samples stored in the Replay Memory. In the second stage, the policy network and target network utilize samples generated in the previous stage to update the policy network through the Q-learning algorithm. The target network is updated periodically after a fixed number of iterations. The two stages are performed alternately to enhance the diagnostic performance of the entire model. A detailed explanation of the learning process is provided below.

1) Fault type identification and sample storing

To assemble samples for learning, the environment first randomly provides a sample $\boldsymbol{X}_n$ composed of flight data. This sample is the input of the fault type identification process. Directly using the agent for identification may lead to an accumulation of erroneous information. To mitigate this, a probability is set for randomly selecting the fault type. This method is called the ϵ-greedy strategy. A value ϵ is set such that when a random number is less than ϵ, a random action is chosen. Otherwise, the policy network is used. At the early stage of training, a larger value of ϵ is set to encourage exploration. As learning progresses, the agent's estimation of Q-values for actions in specific states becomes more precise. Consequently, ϵ is gradually reduced until the agent rarely uses random fault types as classification results. After the identification process ends, the environment provides a reward r based on the comparison between the determined fault type and the actual fault type. Upon receiving a new sample, it is recorded as $\boldsymbol{X}_{n+1}$. This information is stored in the memory, serving as the data source for updating the agent.

2) Network update process

When the data stored in the memory buffer reaches a certain threshold, a learning process is triggered after completing each state identification. The main objective of this learning process is to improve the precision of Q-value estimation for specific states, thereby enhancing fault discrimination capabilities.

The network update process is explained using the learning of a single sample. First, the policy network calculates the Q-value for the current state $\boldsymbol{X}_n$ under the current action a_n. The calculation is given by:

$$Q(\boldsymbol{X}_n, a_n) = \pi(\boldsymbol{X}_n)[a_n] \tag{6}$$

where $\pi(\cdot)$ represents the processing of the policy network. For the next available sample $\boldsymbol{X}_{n+1}$, the policy network calculates the action a_{n+1}. Then, the target network calculates the corresponding Q-value as follows:

$$Q_{n+1}(\boldsymbol{X}_{n+1}) = \pi'(\boldsymbol{X}_{n+1})\left[\mathrm{Argmax}\left(\pi(\boldsymbol{X}_{n+1})\right)\right] \tag{7}$$

where $\pi'(\cdot)$ represents the processing of the target network, and $\mathrm{Argmax}(\cdot)$ denotes

the index of the maximum value in the vector. If $\boldsymbol{X}_{n+1}$ does not exist, Q_{n+1} is directly set to 0. The influence of Q-values during state transitions is calculated using the Bellman equation. The expected Q-value is expressed as:

$$\hat{Q} = r_n + \gamma \cdot Q_{n+1}(\boldsymbol{X}_{n+1}) \tag{8}$$

Here, γ is the discount factor for future rewards in the current Q-value estimation. In fault diagnosis tasks, γ is typically small because the samples provided by the environment lack significant correlation. Finally, the $\hat{Q}$ value is compared with the Q-value from the policy network. The mean squared error is used as the loss function. The policy network is updated through backpropagation. After a certain number of training steps, the parameters of the target network are updated.

3.4 Network Training and Testing

The training process of the overall structure includes the training of the self-encoder, contrastive learning network, and D3QN. During the training of the self-encoder, the learning rate is usually set to a higher value to allow the encoder to fully capture the fundamental patterns of the original samples. During the contrastive learning stage, the encoder's parameters are used as the initial parameters of the contrastive learning network. The learning rate for the encoder in the contrastive learning network is set to a smaller value. Conversely, the projection layer uses a higher learning rate to better distinguish different samples while preserving the fundamental patterns of the data. After training the encoder of the feature distinction module, its parameters are integrated into the type identification module as part of the agent. Subsequently, different rewards are constructed based on the variations in the sample counts of different fault types within the training set, providing feedback to the agent. In this step, the trained encoder's network parameters are frozen, and only the LSTM main network and two linear layers of the agent are trained.

The testing process primarily utilizes the agent. Test samples processed through the same procedure are input into the agent to determine the fault type based on the sample characteristics. The fault types identified by the agent are compared with the actual fault types. This comparison evaluates the performance of different fault diagnosis methods.

4 Experimental results

This section presents the experimental details. First, the dataset used in the experiment is introduced, followed by a description of the experimental platform and the hyperparameters configured for each module. Second, comparative experiments are conducted among multiple methods to evaluate their performance from various perspectives. Finally, the impact of various modules of CCRL is assessed through ablation experiments.

4.1 Experimental setup

4.1.1 Detail of the dataset

Fig. 5 illustrates the overall structure of the CFM56-7B series turbofan engine and the associated data collection, transmission, and conversion processes. The engine

consists of key components, including the High-Pressure Compressor (HPC), Low-Pressure Compressor (LPC), High-Pressure Turbine (HPT), Low-Pressure Turbine (LPT), and Combustion Chamber (CC). As indicated in Fig. 5, critical gas path performance parameters such as N_1, N_2, FF and EGT, are collected by sensors installed near the low-pressure compressor, high-pressure compressor, combustion chamber, and low-pressure turbine regions. These parameters, widely used in gas path monitoring[33]. To monitor engine performance, civil aviation engines rely on air-ground communication systems like ACARS, which transmit real-time operational data at critical flight phases (e.g., takeoff and landing). However, airlines typically receive only raw sensor readings, which require further processing to support maintenance and fault diagnosis. Compared to airlines, OEMs possess deeper insights into engine performance. By leveraging proprietary algorithms, OEMs refine raw parameters into more informative metrics, improving diagnostic accuracy. The dataset used in this study integrates these refined parameters with original sensor data and incorporates fault annotations derived from the Condition Notification Report, creating a more comprehensive dataset for fault diagnosis.

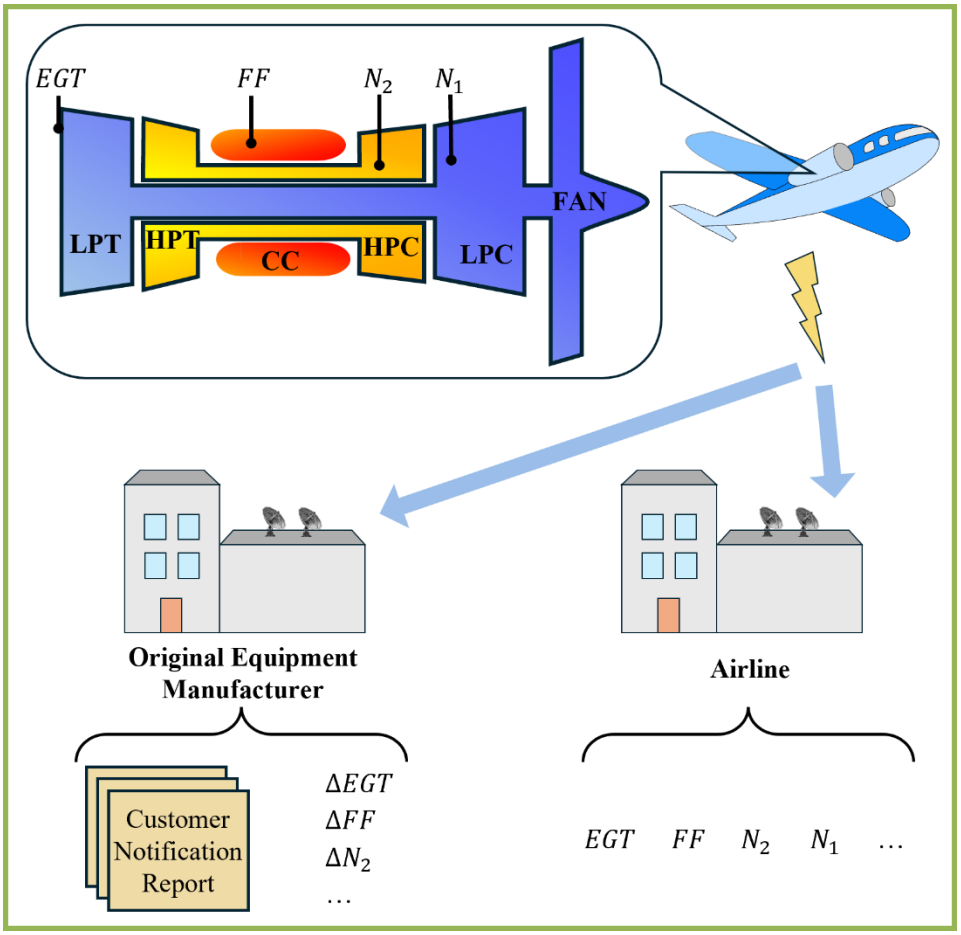

Fig. 5. The structure of the aero-engine with its sensor data collection and conversion process.

In addition to normal data, the dataset used in this experiment includes three types of faults: (1) VBV system failure, which can result in loss of control over the compressor airflow, potentially leading to engine stalls or surges. (2) EGTI, which can

lead to engine overheating or a reduction in performance. (3) TAT sensor failure. The total temperature sensor measures air temperature to adjust the fuel injectors. When the sensor malfunctions, it can result in inaccurate fuel supply, adversely affecting engine thrust. The normal, VBV, EGTI, and TAT classes have fifty, twenty-one, fourteen, and eighteen training samples, respectively; the numbers of testing samples of these classes are all ten. The reason is that due to practical constraints, only limited data is available for testing the proposed method before formal deployment.

Fig. 6 illustrates the data patterns of a single aero-engine and the source of fault diagnosis samples. The sensor data sequence for each flight, denoted as $\boldsymbol{x}_m$, forms a complete record with a length of L. In the experiment, for a specific fault type, the first M flights observed before a fault occurrence are selected as samples for the fault type. The remaining data are treated as samples for the normal type, thus forming the actual dataset. To distinguish faulty and normal samples, it was noted that most failure flight durations in the customer notification report do not exceed 10 [34]. Therefore, the sample length M and sliding window size were both set to 10. During data collection, sampling begins from the flight where the fault occurs. A sample of a specific fault type can be represented as $\boldsymbol{X}_n = \{\boldsymbol{x}_{L-9}, \boldsymbol{x}_{L-8}, \boldsymbol{x}_{L-7}, \dots, \boldsymbol{x}_{L-1}, \boldsymbol{x}_L\}$, which is a matrix combining data from the last 10 flights. For normal samples, 60 non-overlapping groups of data are selected from $M - 10$ normal flight data.

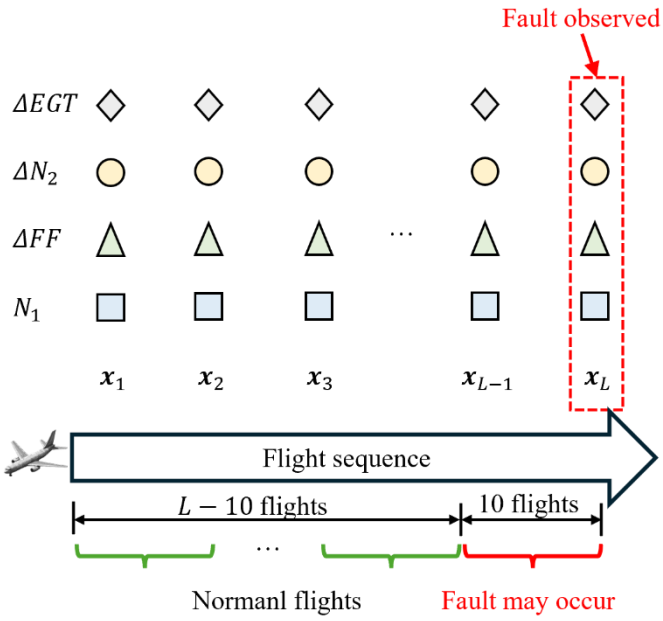

Fig. 6. Flight data sampling process and dataset generation.

4.1.2 Experimental setup

The proposed method consists of multiple modules. The following hyperparameters are specified for each module, and the hyperparameters of the same module are consistent across different experiments.

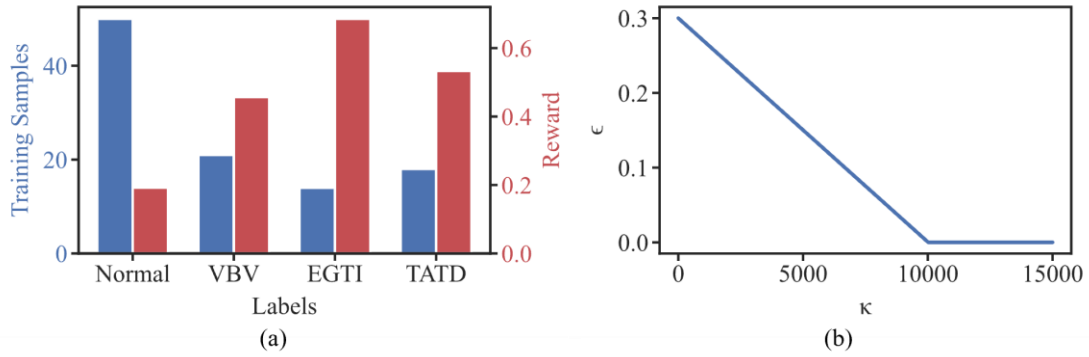

Fig. 7. Training-related hyperparameter design. (a) Relationship between the number of training samples and reward. (b) Relationship between κ and the exploration-exploitation threshold ϵ during training.

The LSTM network serves as the primary network of the D3QN agent. The input dimension of CCRL is set to 32, while it is 4 for the other methods. The hidden layer dimension is 32, and the network consists of a single layer. During training, γ is set to 0.01 due to the relative independence of each training process. The memory size is set to 80, and training begins when the number of samples in memory exceeds 50. The target network is updated every 500 training iterations. The learning rate is set to 0.001, the batch size is 16, and the maximum number of episodes is 300. Due to the small size of the dataset, no validation set is used. The early stopping strategy is applied when the total loss on the training set does not decrease for 10 consecutive episodes. In the D3QN experiment, the fault diagnosis reward for all four fault types is set to 1. In the CCRL experiment, the reward distributions for different training sample quantities are shown in Fig. 7(a). The calculation rule for ϵ is as follows:

$$\epsilon(\kappa) = \begin{cases} \epsilon_s + (\epsilon_e - \epsilon_s) \times \kappa/\kappa_{\max} & \text{if } \kappa < \kappa_{\max} \\ \epsilon_e & \text{if } \kappa \geq \kappa_{\max} \end{cases} \tag{9}$$

Here, κ represents the learning step. ϵ_s denotes the initial value of ϵ, set to 0.3, ϵ_e is the final value of ϵ, set to 0.0001. $\kappa_{\max}$ represents the maximum learning step, set to 10000. The variation of ϵ during training is shown in Fig. 7(b). Here, ϵ balances exploration and exploitation in ϵ-greedy policy. Initially, a large ϵ ensures sufficient exploration to discover optimal Q-values. As the agent gains experience, ϵ decreases to prioritize learned policies. Once the network stabilizes, maintaining a small ϵ prevents premature convergence. The batch size for each learning iteration is 16.

For the LSTM-based autoencoder, the maximum number of epochs is set to 500. Training stops early if the loss does not decrease for 10 consecutive epochs. The learning rate is 0.001. The LSTM input dimension is 4, and the hidden layer size is 32. The encoder of the LSTM autoencoder is directly used in the contrastive learning network. The input to the projection layer for contrastive learning is 320, the hidden layer size is 160, and the output layer size is 4. Based on grid search, the temperature parameter τ for the loss function in contrastive learning is set to 0.3. The weight for positive sample pairs w_p is 0.8, and the weight for negative sample pairs w_n is 0.3. For the LSTM part, the learning rate is set to 0.0001, and for the projection part, it is

set to 0.001. The maximum number of epochs for training is 500. Early stopping is applied if the training loss does not decrease for 10 consecutive epochs.

4.2 Analysis of experimental results

The proposed method CCRL is compared with three baseline methods: (1) D3QN without any processing during training, (2) Down Sampler (DS) + D3QN, and (3) Over Sampler (OS) + D3QN. “DS” refers to the approach where the fault type with the fewest samples is identified, and data from fault types with more samples are randomly deleted to ensure an equal number of training samples for each fault type. “OS” involves identifying the fault type with the largest number of samples and replicating the samples from other fault types until the number of samples for all fault types matches the largest sample count. To evaluate the performance of the model, training and test samples are randomly allocated, and the experiment is repeated 10 times to provide more reliable performance estimates.

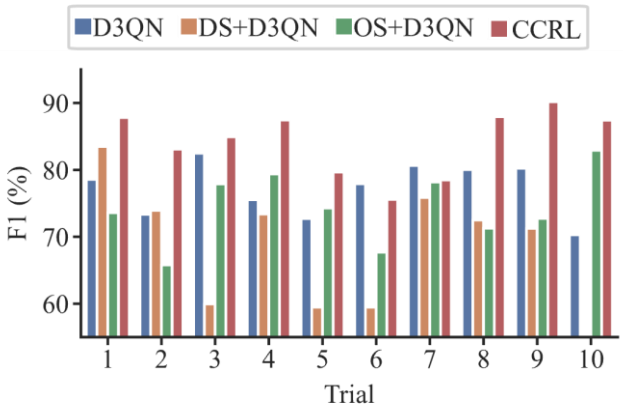

Fig. 8. F1 Scores of four methods across 10 repeated experiments.

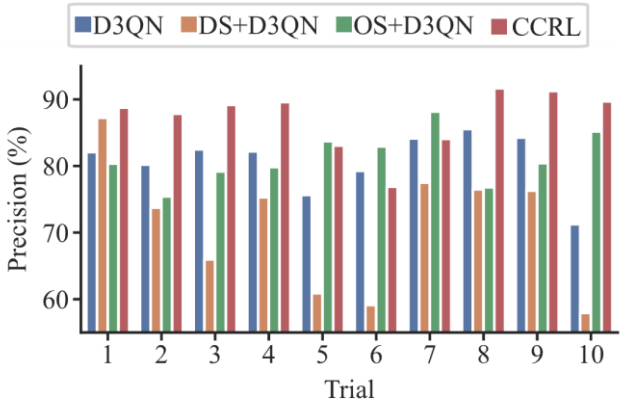

Fig. 9. Precision of four methods across 10 repeated experiments.

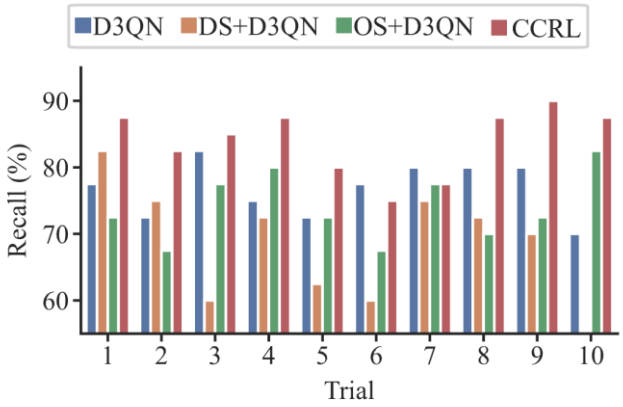

Fig. 10. Recall of four methods across 10 repeated experiments.

Fig. 8-Fig. 10 illustrate the F1 score, precision, and recall results of the experiments, where the F1 score represents the harmonic mean of precision and recall. As shown in Fig. 8, the overall performance of OS+D3QN surpasses that of DS+D3QN. This improvement may be attributed to the better utilization of available training data.

Among the four methods, DS+D3QN exhibits the worst performance, while CCRL achieves the best performance. In 10 repeated experiments, CCRL outperformed other methods in 8 instances, indicating the effectiveness of the proposed method. In the 6th and 7th experiments, the performance of CCRL was worse than that of D3QN. This may be due to the random allocation of the dataset, causing the test set to fail in representing the general characteristics of various faults.

Table 1 Average and standard deviation of experimental results (%) for different methods.

Average metric	Method			
	D3QN	DS+D3QN	OS+D3QN	CCRL
F1	77.19 ± 3.81	68.15 ± 9.26	74.38 ± 5.07	**84.26 ± 4.62**
Precision	80.71 ± 4.15	71.05 ± 9.10	81.20 ± 3.66	**87.19 ± 4.34**
Recall	76.75 ± 3.88	68.00 ± 9.14	74.00 ± 4.90	**84.00 ± 4.77**

Table 1 summarizes the average and standard deviation of the evaluation metrics presented in Fig. 8-Fig. 10. As shown in Table 1, CCRL achieves the best performance in the three metrics, outperforming the compared methods by 9.16%, 7.38%, and 9.45%, respectively. CCRL effectively improves fault diagnosis performance for aero-engine under high imbalance without requiring an increase in sample quantity.

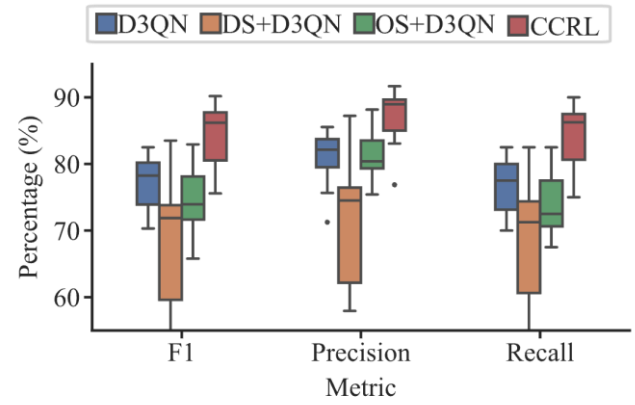

Fig. 11. Experimental results of four methods across 10 repeated experiments.

The box plot in Fig. 11 provides further insights into the differences and stability of these methods. Based on the performance of the F1 score, precision, and recall, CCRL shows good stability and achieves high median values across all three metrics. Specifically, the median F1 score of CCRL is significantly higher than that of the comparison methods, indicating that CCRL better balances precision and recall. The results of DS+D3QN show a relatively long range, indicating greater variability. The whiskers of the box plot further reveal the dispersion of its F1 scores. Similarly, DS+D3QN exhibits the highest variability among the four methods in terms of precision. In terms of recall, the other three methods generally achieve lower scores compared to CCRL.

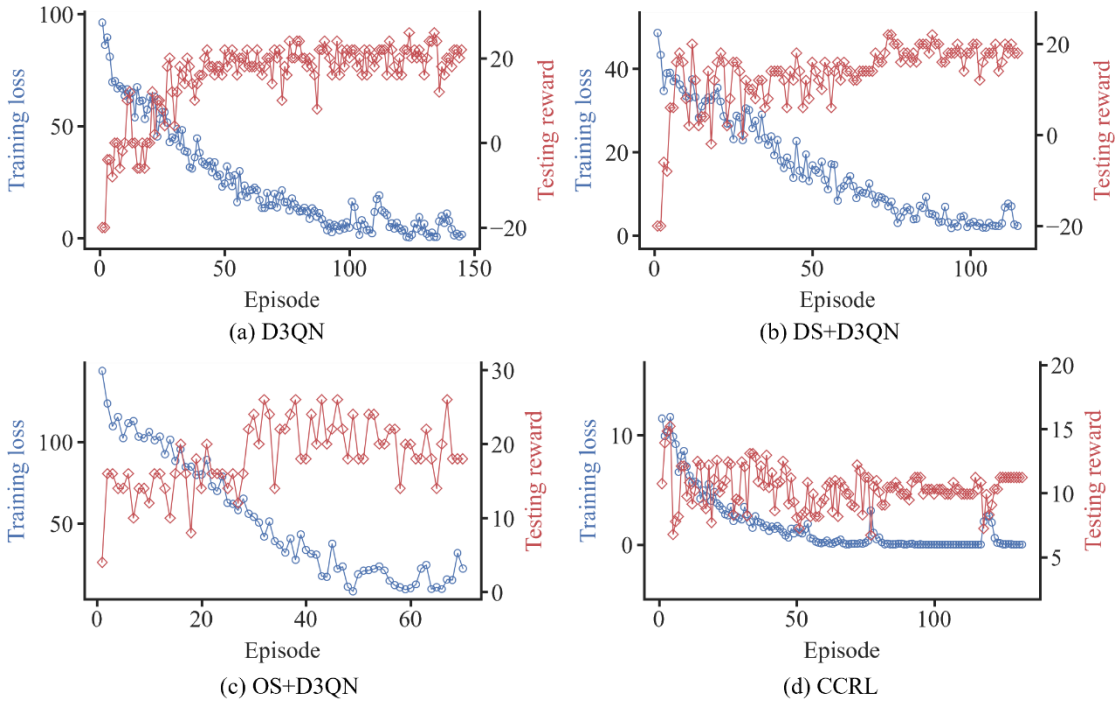

Fig. 12. Changes in training loss and test rewards during the training stage.

Fig. 12 illustrates the training loss and test rewards of the four methods proposed in this paper after each episode during the training process. The plot for each method represents the result closest to the average F1 score from 10 repeated experiments and surpasses the average F1 score. Based on the above conditions, D3QN, DS+D3QN, OS+D3QN, and CCRL use the results of the 6th, 8th, 5th, and 3rd repetitions, respectively, to represent the average performance of each method. Training loss is defined as the total loss during learning when the number of samples in memory reaches the minimum allowable threshold. Test rewards refer to the final reward obtained by the policy network interacting with the environment after each training episode. Since the rewards for the four experimental groups are not identical, the evaluation of each method can only be performed based on a qualitative analysis of their patterns. The first episode is excluded as the network had not begun learning or generating losses initially.

In Fig. 12(a), D3QN exhibits rapid convergence during the first 100 episodes, with the total loss approaching 0 around the 100th episode. The test rewards increase to 20 after 50 episodes and then fluctuate within the range of 10 to 20. Fig. 12(b) presents the experimental results of DS+D3QN. The training loss decreases to 0 and stabilizes after 100 episodes. The test rewards initially increase rapidly and then gradually stabilize as training progresses. Fig. 12(c) illustrates the results of OS+D3QN. The training loss stabilizes around the 60th episode. However, the test rewards exhibit significant fluctuations between 15 and 25, and the latter part of the training process remains unstable. Fig. 12(d) shows the training process of CCRL, characterized by a consistent decline in training loss, which continues to improve near 0. Regarding test rewards, they gradually stabilize at higher levels as the training progresses, showing superior stability. Overall, all methods have achieved convergence on the training set.

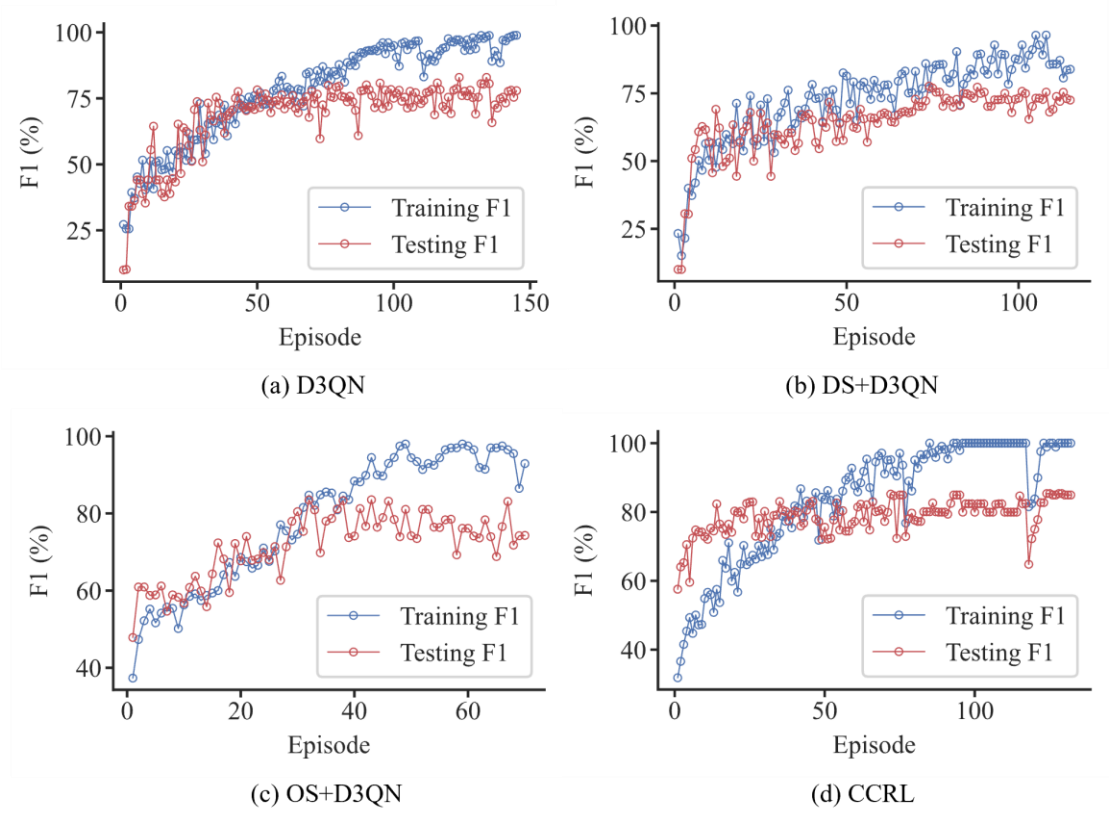

Fig. 13. Changes in training F1 and test F1 during the training stage.

Fig. 13 illustrates the F1 scores of four methods on training and test samples during the training process. In terms of training F1, the processes depicted in Fig. 13(a) and Fig. 13(d) are relatively smooth and eventually stabilize near 100%. In contrast, the training processes in Fig. 13(b) and Fig. 13(c) fail to stabilize at 100%. In terms of test F1, the scores of D3QN and DS+D3QN stabilize at 75%. Although the test F1 of OS+D3QN reaches 80%, it fails to maintain this level. In contrast, the test F1 of CCRL increases rapidly during the early stages and stabilizes at 80% later. These results indicate that CCRL outperforms the other three methods. However, across all four training processes, the F1 scores for both the training and test sets exhibit varying degrees of overfitting. This phenomenon is likely due to the limited number of training samples, which represents an inherent challenge in fault diagnosis tasks involving scarce fault samples.

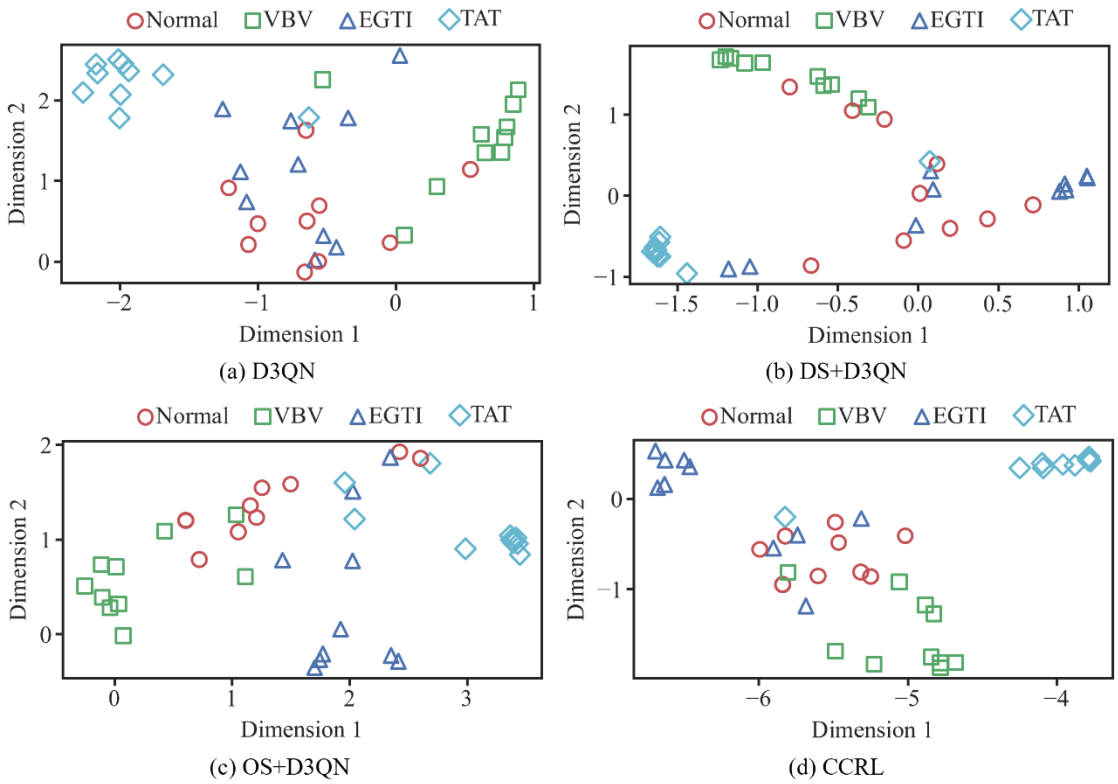

Fig. 14. Distribution of test samples in two-dimensional feature space.

Fig. 14 illustrates the use of t-SNE to project the high-dimensional features of the final output from the D3QN module of each method into a two-dimensional space. As shown in the four subplots, the fault types VBV and TAT are relatively easier to distinguish, as their distributions in the two-dimensional feature space are more concentrated. The distributions of the EGTI and normal fault types appear more chaotic. Except for CCRL, the normal fault type features are relatively scattered, particularly in the DS+D3QN method. This may be because the DS operation reduces a large number of normal samples, leading to an incomplete representation of their data distribution. In contrast, the feature distributions of different types in the CCRL sample diagram are relatively compact, which further verifies CCRL's potential for discriminative feature learning.

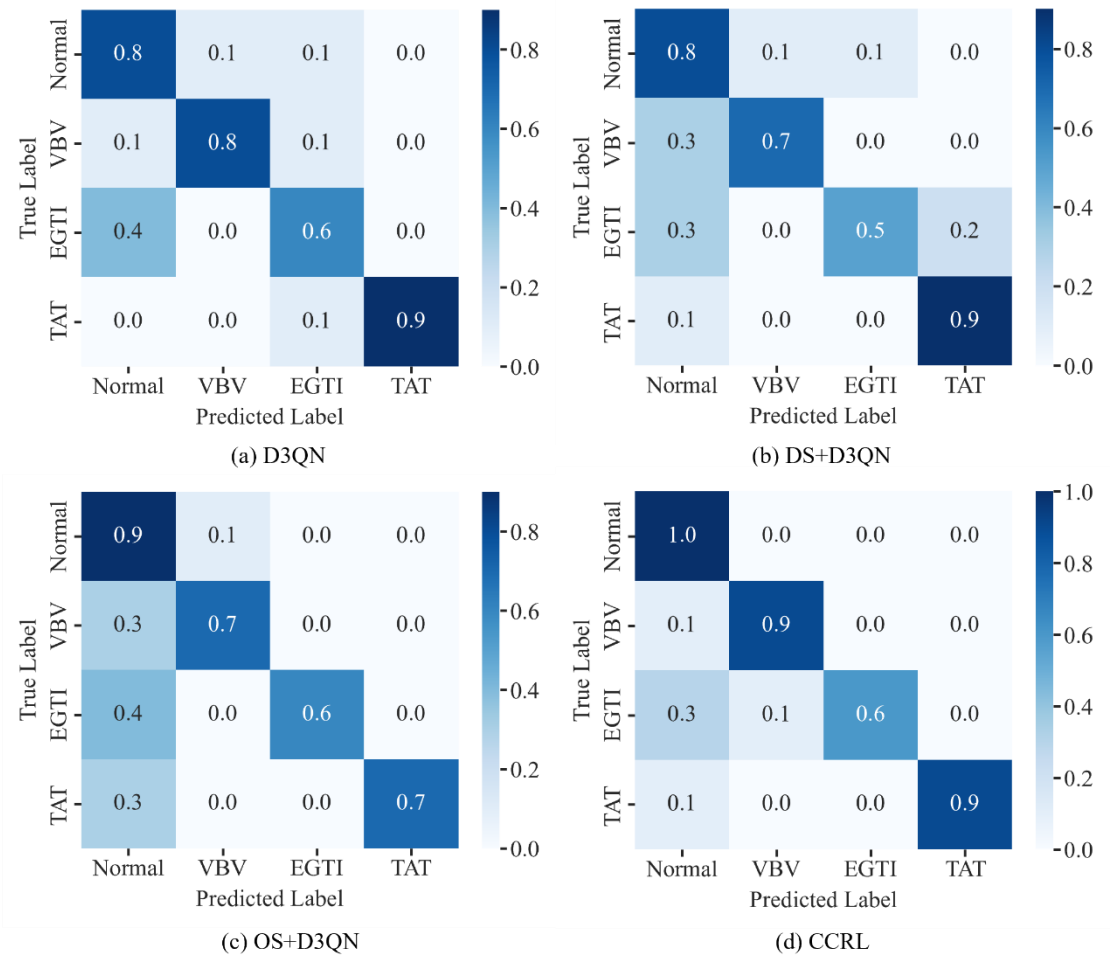

Fig. 15. Confusion matrices of four methods on the test set.

In Fig. 15, the confusion matrices of the four methods provide further insights into their performance in distinguishing different fault types. In summary, several methods misclassify samples of various fault types as normal. This may be attributed to the diverse patterns exhibited by normal samples, which contribute to the higher classification precision for this class but also result in the misclassification of several fault samples as normal. In addition, TAT achieves higher overall precision among the four fault types, while EGTI is more challenging to distinguish, likely due to the small size of the EGTI dataset. Compared with other methods, CCRL does not significantly improve the precision for EGTI but performs well in distinguishing TAT and VBV fault types. In addition, all normal samples are correctly classified.

4.3 Analysis of ablation experiment results

The effectiveness of various modules of CCRL is evaluated through ablation experiments. The ablation experiments include: (1) D3QN with rewards adjusted based on the number of training samples (abbreviated as I-D); (2) D3QN with rewards adjusted using features pre-processed by an autoencoder (AE) (abbreviated as A-I-D); (3) D3QN with unbalanced rewards applied after training a contrastive learning network (abbreviated as C-I-D); and (4) D3QN with features derived from a contrastive

learning network pre-trained with an autoencoder (abbreviated as A-C-D). The four methods in the ablation experiments are constructed by integrating one or more modules of CCRL, allowing the contribution of each module to the overall performance to be observed.

Table 2 presents the results of the training and test sets from 10 experiments conducted as described above. The F1 measurements indicate that the performance of the split methods is lower than that of CCRL, with a larger standard deviation observed across the 10 repeated experiments. Among these methods, the best performance is achieved by the D3QN which utilizes features from a contrastive learning network pre-trained with an autoencoder. This indicates that for fault diagnosis tasks, effective feature distinction is more critical than sample balance. Compared to CCRL, the removal of the imbalanced sample design increased the standard deviation of ten repeated experiments from 4.62 to 5.60. Among these methods, A-C-D exhibits the smallest F1 standard deviation, indicating that the contrastive learning module designed in this paper effectively stabilizes the results of repeated experiments. In terms of precision and recall, A-C-D outperforms the other three methods in the ablation experiments but still falls short compared to CCRL. In summary, the coordination of multiple modules in CCRL achieves a higher average performance and smaller standard deviation in the experiment.

Table 2 Average and standard deviation (%) of experimental results for the four methods in ablation experiments.

Average metric	Method			
	I-D	A-I-D	C-I-D	A-C-D
F1	75.17 ± 6.63	77.78 ± 7.70	74.13 ± 7.10	79.47 ± 5.60
Precision	78.36 ± 5.99	80.35 ± 7.59	76.83 ± 5.69	84.72 ± 3.67
Recall	75.75 ± 5.81	77.25 ± 7.86	74.25 ± 7.25	79.00 ± 5.72

In the 10 repeated experiments, the experiment index with the F1 score closest to the average F1 and greater than the average F1 was selected. Partial output features of D3QN were visualized using t-SNE dimensionality reduction, as shown in Fig. 16. The data for I-D, A-I-D, C-I-D, and A-C-D were obtained from the 5th, 7th, 3rd, and 1st experiments, respectively, in their corresponding 10 repeated experiments. Among the subplots of the four methods, Fig. 16(a) shows that the features of normal, VBV, and EGTI faults are not effectively distinguished, especially between normal and VBV. This indicates that the autoencoder combined with contrastive learning can effectively reduce the dispersion of different fault types. By comparing Fig. 16(b) and Fig. 16(c), it is evident that Fig. 16(c) achieves a better distinction between normal and VBV. Therefore, the reduction in dispersion among different fault types is mainly attributed to the contrastive learning process. In Fig. 16(d), VBV and TAT are distinguished, and there is a certain degree of distinction for the other two fault types as well.

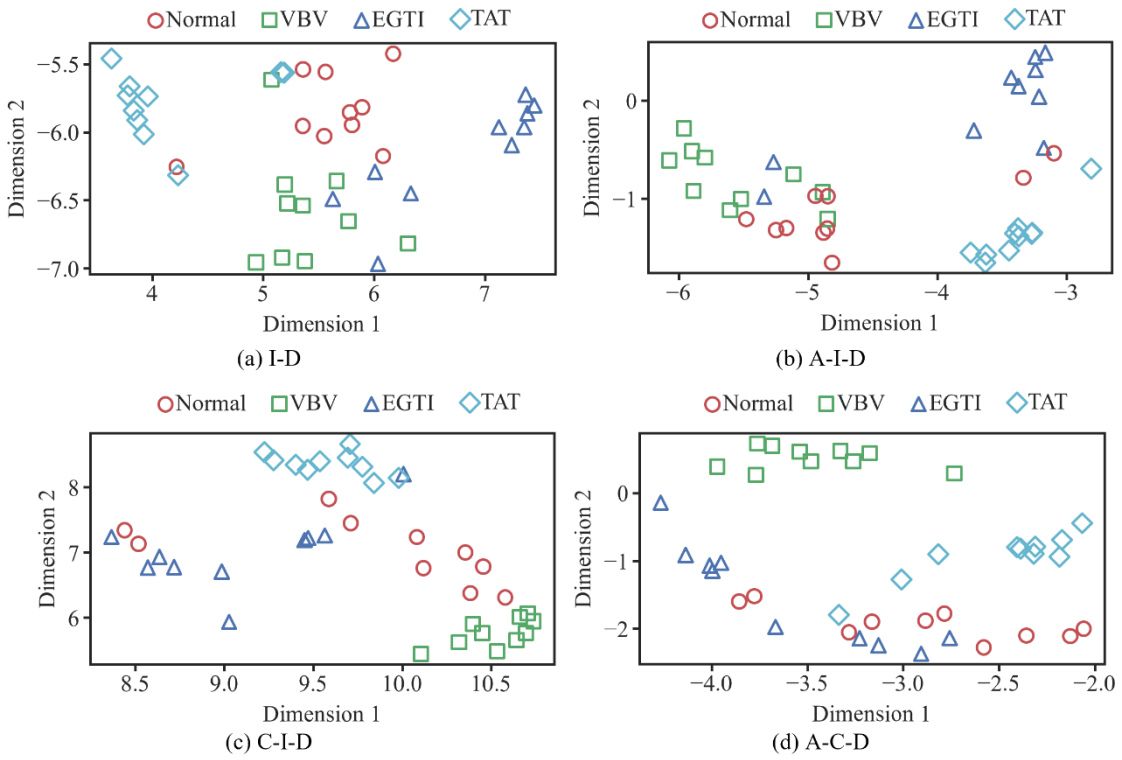

Fig. 16. Distribution of test samples in two-dimensional feature space in ablation experiment.

Fig. 17 presents the confusion matrix results of the four methods used in the ablation experiments. As shown in Fig. 17(a), (c), and (d), the EGTI fault type is the most affected by the imbalance in the training samples due to the minimal number of EGTI samples in the training set. For VBV and TAT, which are easier to distinguish, Fig. 17(c) and Fig. 17(d) show better results. Additionally, A-C-D shows better distinction for the normal type compared to C-I-D. This indicates that in fault diagnosis, training effective contrastive networks is more beneficial than only addressing sample imbalance.

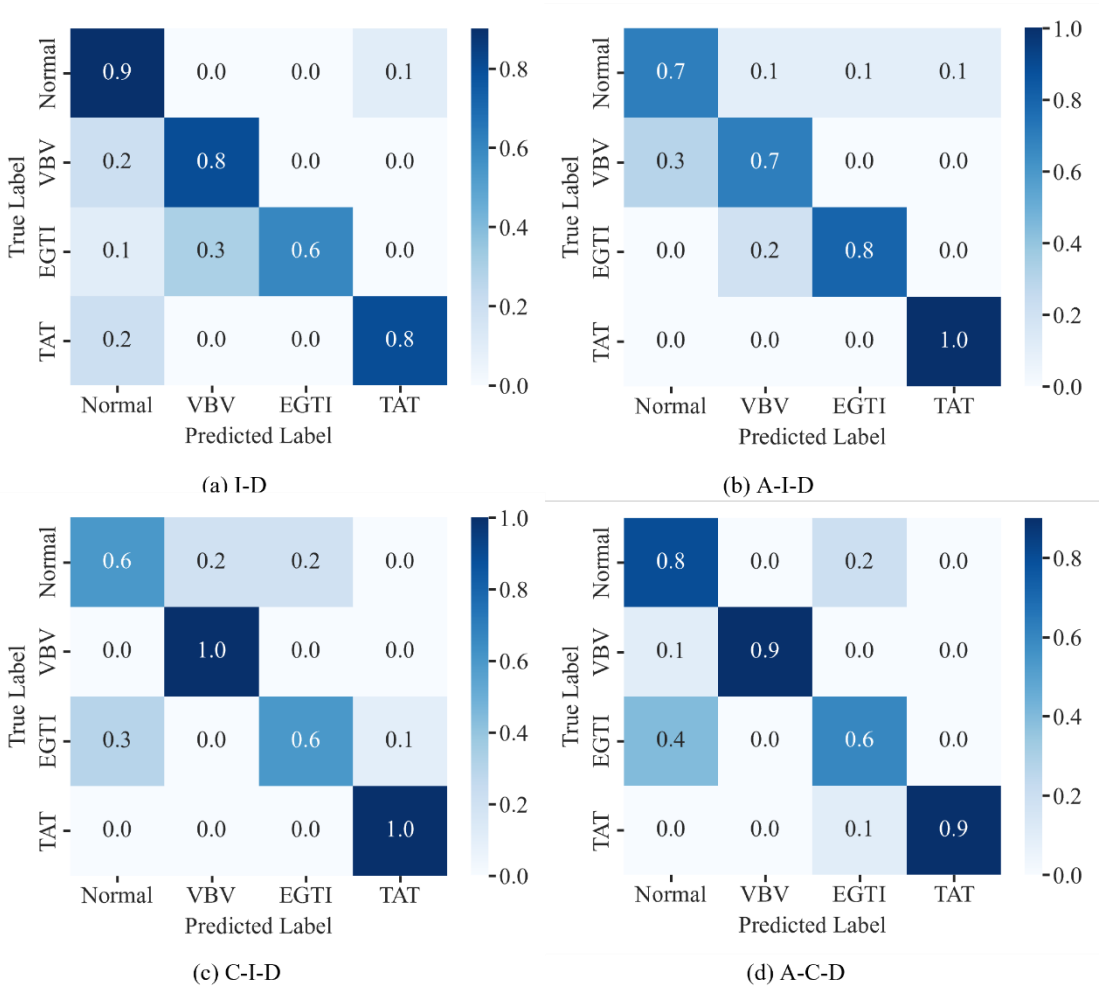

Fig. 17. Confusion matrices of four methods in ablation experiment.

5 Conclusion

This paper proposes a machine learning method with environmental interaction and continual dynamic evolution capabilities, referred to as Continual Contrastive Reinforcement Learning. This method treats the operating environment of the airline as the learning environment for the agent. Using in-flight status data and expert fault judgments, the agent can be continually trained to enhance its ability to adapt to environmental changes. The framework consists of feature distinction modules and type identification modules. The feature distinction module is designed to address the challenge of having a wide variety of fault types but limited samples for each type. Original data is transformed into positive and negative samples. The contrastive learning encoder is pre-trained using an autoencoder, and the contrastive learning loss function is refined to account for the imbalance between positive and negative sample quantities. The type identification module addresses the issue of sample imbalance. The trained encoder from the feature distinction module is embedded into the agent of the type identification module. Rewards are assigned based on the number of samples of different fault types to mitigate the impact of sample imbalance during training. This study conducts two sets of experiments using real aero-engine data. The comparative

experiments show the effectiveness of the proposed framework, while the ablation experiments show that the design of each module aligns with the intended objectives and performs as expected.

This framework is not only suitable for fault diagnosis of aero-engines just using available fault samples, but also continually upgradable along with the increasing data amount. It is also applicable to fault diagnosis of other complex equipment under highly imbalance scenarios. Beyond that, the constructed feature distinction module can serve as the feature distinction module for other classification methods as well.

In future work, we plan to extend the number of fault types that the agent can diagnose and explore zero-shot fault diagnosis. Additionally, we aim to better integrate Quick Access Recorder and ACARS data to provide more comprehensive data support for fault diagnosis. Moreover, we are also interested in investigating how to optimize the collaboration between the "Feature Distinction Module" and the "Type Identification Module" to better align with the development of airline operations.

6 Acknowledgment

This work was supported by National Key R&D Program of China (2023YFB4302400).

7 Reference

[1] M. Hakim, A.A.B. Omran, A.N. Ahmed, M. Al-Waily, A. Abdellatif, A systematic review of rolling bearing fault diagnoses based on deep learning and transfer learning: Taxonomy, overview, application, open challenges, weaknesses and recommendations, Ain Shams Engineering Journal 14 (2023) 101945. https://doi.org/10.1016/j.asej.2022.101945.

[2] H. Shi, S. Cao, H. Zuo, J. Ma, C. Lin, Deep subdomain adversarial network with self-supervised learning for aero-engine high speed bearing fault diagnosis with unknown working conditions, Measurement 241 (2025) 115668. https://doi.org/10.1016/j.measurement.2024.115668.

[3] M. Chen, R. Qu, W. Fang, Case-based reasoning system for fault diagnosis of aero-engines, Expert Systems with Applications 202 (2022) 117350. https://doi.org/10.1016/j.eswa.2022.117350.

[4] X. Liu, Y. Chen, L. Xiong, J. Wang, C. Luo, L. Zhang, K. Wang, Intelligent fault diagnosis methods toward gas turbine: A review, Chinese Journal of Aeronautics 37 (2024) 93–120. https://doi.org/10.1016/j.cja.2023.09.024.

[5] Z. Tan, S. Zhong, L. Lin, A model learning strategy adapted to health assessment of multi-component systems, in: 2017 Prognostics and System Health Management Conference (PHM-Harbin), 2017: pp. 1–7. https://doi.org/10.1109/PHM.2017.8079223.

[6] H. Zhang, X. Chen, W. Chen, Z. Shen, Collaborative sparse classification for aero-engine's gear hub crack diagnosis, Mechanical Systems and Signal Processing 141 (2020) 106426. https://doi.org/10.1016/j.ymssp.2019.106426.

[7] J. Hu, M. Chen, H. Tang, J. Zhang, An adversarial transfer learning method based on domain distribution prediction for aero-engine fault diagnosis, Engineering

Applications of Artificial Intelligence 133 (2024) 108287. https://doi.org/10.1016/j.engappai.2024.108287.

[8] Y. Lv, W. Zhao, Z. Zhao, W. Li, K.K.H. Ng, Vibration signal-based early fault prognosis: Status quo and applications, Advanced Engineering Informatics 52 (2022) 101609. https://doi.org/10.1016/j.aei.2022.101609.

[9] Z. Dongzhu, Z. Hua, D. Shiqiang, S. Yafei, Aero-engine Bearing Fault Diagnosis Based on Deep Neural Networks, in: 2020 11th International Conference on Mechanical and Aerospace Engineering (ICMAE), 2020: pp. 145–149. https://doi.org/10.1109/ICMAE50897.2020.9178886.

[10] L. Wen, X. Li, L. Gao, A New Two-Level Hierarchical Diagnosis Network Based on Convolutional Neural Network, IEEE Transactions on Instrumentation and Measurement 69 (2020) 330–338. https://doi.org/10.1109/TIM.2019.2896370.

[11] J. Liu, C. Pan, F. Lei, D. Hu, H. Zuo, Fault prediction of bearings based on LSTM and statistical process analysis, Reliability Engineering & System Safety 214 (2021) 107646. https://doi.org/10.1016/j.ress.2021.107646.

[12] H. Lee, H. Jeong, G. Koo, J. Ban, S.W. Kim, Attention Recurrent Neural Network-Based Severity Estimation Method for Interturn Short-Circuit Fault in Permanent Magnet Synchronous Machines, IEEE Trans. Ind. Electron. 68 (2021) 3445–3453. https://doi.org/10.1109/TIE.2020.2978690.

[13] T. Pan, J. Chen, T. Zhang, S. Liu, S. He, H. Lv, Generative adversarial network in mechanical fault diagnosis under small sample: A systematic review on applications and future perspectives, ISA Transactions 128 (2022) 1–10. https://doi.org/10.1016/j.isatra.2021.11.040.

[14] L. Meng, M. Zhao, Z. Cui, X. Zhang, S. Zhong, Empirical mode reconstruction: Preserving intrinsic components in data augmentation for intelligent fault diagnosis of civil aviation hydraulic pumps, Computers in Industry 134 (2022) 103557. https://doi.org/10.1016/j.compind.2021.103557.

[15] S. Fu, L. Lin, Y. Wang, M. Zhao, F. Guo, S. Zhong, Y. Liu, High imbalance fault diagnosis of aviation hydraulic pump based on data augmentation via local wavelet similarity fusion, Mechanical Systems and Signal Processing 209 (2024) 111115. https://doi.org/10.1016/j.ymssp.2024.111115.

[16] K. Zhang, Q. Wen, C. Zhang, R. Cai, M. Jin, Y. Liu, J.Y. Zhang, Y. Liang, G. Pang, D. Song, S. Pan, Self-Supervised Learning for Time Series Analysis: Taxonomy, Progress, and Prospects, IEEE Transactions on Pattern Analysis and Machine Intelligence 46 (2024) 6775–6794. https://doi.org/10.1109/TPAMI.2024.3387317.

[17] K. Kamycki, T. Kapuscinski, M. Oszust, Data Augmentation with Suboptimal Warping for Time-Series Classification, Sensors 20 (2020) 98. https://doi.org/10.3390/s20010098.

[18] D. Liu, S. Zhong, L. Lin, M. Zhao, X. Fu, X. Liu, Deep attention SMOTE: Data augmentation with a learnable interpolation factor for imbalanced anomaly detection of gas turbines, Computers in Industry 151 (2023) 103972. https://doi.org/10.1016/j.compind.2023.103972.

[19] H. Lu, M. Du, K. Qian, X. He, K. Wang, GAN-Based Data Augmentation Strategy for Sensor Anomaly Detection in Industrial Robots, IEEE Sensors Journal 22

(2022) 17464–17474. https://doi.org/10.1109/JSEN.2021.3069452.
[20] Iwana B.K., Uchida S., An empirical survey of data augmentation for time series classification with neural networks, PLOS ONE 16 (2021) e0254841. https://doi.org/10.1371/journal.pone.0254841.
[21] J. He, M. Ouyang, Z. Chen, D. Chen, S. Liu, A Deep Transfer Learning Fault Diagnosis Method Based on WGAN and Minimum Singular Value for Non-Homologous Bearing, IEEE Transactions on Instrumentation and Measurement 71 (2022) 1–9. https://doi.org/10.1109/TIM.2022.3160533.
[22] S. Zhong, S. Fu, L. Lin, A novel gas turbine fault diagnosis method based on transfer learning with CNN, Measurement 137 (2019) 435–453. https://doi.org/10.1016/j.measurement.2019.01.022.
[23] W. Chen, Y. Qiu, Y. Feng, Y. Li, A. Kusiak, Diagnosis of wind turbine faults with transfer learning algorithms, Renewable Energy 163 (2021) 2053–2067. https://doi.org/10.1016/j.renene.2020.10.121.
[24] J. Liu, Gas path fault diagnosis of aircraft engine using HELM and transfer learning, Engineering Applications of Artificial Intelligence 114 (2022) 105149. https://doi.org/10.1016/j.engappai.2022.105149.
[25] C. Martinez, G. Perrin, E. Ramasso, M. Rombaut, A Deep Reinforcement Learning Approach for Early Classification of Time Series, in: 2018 26th European Signal Processing Conference (EUSIPCO), 2018: pp. 2030–2034. https://doi.org/10.23919/EUSIPCO.2018.8553544.
[26] Y. Ding, L. Ma, J. Ma, M. Suo, L. Tao, Y. Cheng, C. Lu, Intelligent fault diagnosis for rotating machinery using deep Q-network based health state classification: A deep reinforcement learning approach, Advanced Engineering Informatics 42 (2019) 100977. https://doi.org/10.1016/j.aei.2019.100977.
[27] E. Lin, Q. Chen, X. Qi, Deep reinforcement learning for imbalanced classification, Appl. Intell. 50 (2020) 2488–2502. https://doi.org/10.1007/s10489-020-01637-z.
[28] X. Wang, S. Yang, J. Zhang, M. Wang, J. Zhang, W. Yang, J. Huang, X. Han, Transformer-based unsupervised contrastive learning for histopathological image classification, Medical Image Analysis 81 (2022) 102559. https://doi.org/10.1016/j.media.2022.102559.
[29] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, P. Bojanowski, DINOv2: Learning Robust Visual Features without Supervision, (2024). https://doi.org/10.48550/arXiv.2304.07193.
[30] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A Simple Framework for Contrastive Learning of Visual Representations, in: International Conference on Machine Learning, PMLR, 2020: pp. 1597–1607. https://proceedings.mlr.press/v119/chen20j.html (accessed October 15, 2024).
[31] Z. Li, S.-S. Zhong, L. Lin, Novel Gas Turbine Fault Diagnosis Method Based on Performance Deviation Model, Journal of Propulsion and Power (2016). https://doi.org/10.2514/1.B36267.

[32] X. Yang, Z. Zhang, R. Cui, TimeCLR: A self-supervised contrastive learning framework for univariate time series representation, Knowledge-Based Systems 245 (2022) 108606. https://doi.org/10.1016/j.knosys.2022.108606.
[33] R. Ganguli, Jet Engine Gas-Path Measurement Filtering Using Center Weighted Idempotent Median Filters, Journal of Propulsion and Power (2012). https://doi.org/10.2514/2.6186.
[34] D. Liu, S. Zhong, L. Lin, M. Zhao, X. Fu, X. Liu, Highly imbalanced fault diagnosis of gas turbines via clustering-based down sampling and deep siamese self-attention network, Advanced Engineering Informatics 54 (2022) 101725. https://doi.org/10.1016/j.aei.2022.101725.