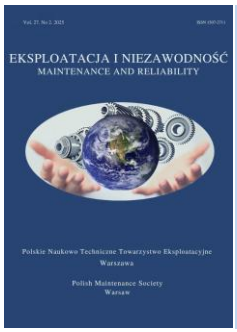


Eksploatacja i Niezawodnosc – Maintenance and Reliability
Volume 27 (2025), Issue 2
journal homepage: http://www.ein.org.pl

Article citation info:
Lv Y, Zhou N, Wen Z, Shen Z, Chen A, Remaining Useful Life Prediction based on Multisource Domain Transfer and Unsupervised Alignment, Eksploatacja i Niezawodnosc – Maintenance and Reliability 2025: 27(2) http://doi.org/10.17531/ein/194116

Remaining Useful Life Prediction based on Multisource Domain Transfer and Unsupervised Alignment

Indexed by:
Web of Science Group

Yi Lv[a,b,*], Ningxu Zhou[b], Zhenfei Wen[b], Zaichen Shen[c], Aiguo Chen[b]

[a] School of Computer, University of Electronic Science and Technology of China, Zhongshan Institute, China
[b] School of Computer Science and Engineering, University of Electronic Science and Technology of China, China
[c] Guangdong University of Technology, China

Highlights

- A generalizable model based on multisource domain transfer RUL prediction is proposed.
- An adaptive alignment mechanism is proposed for feature alignment.
- The prediction model combines the TCN and DANN to make full use of the timing information in the degradation data.

Abstract

Transfer learning enhances remaining useful life (RUL) predictions by addressing data scarcity and operational challenges. Nonetheless, when a significant disparity in degradation data distribution exists between source and target domains, single-source domain transfer learning risks misleading or negative transfer. Multisource domain transfer learning partially addresses these issues. However, it ignores substantial discrepancies in feature-label correlations, which would impair the RUL prediction accuracy. Thus, we propose to develop a multisource domain unsupervised adaptive learning method, which is powered by a temporal convolutional network. Using a multilinear conditioning strategy, we combine degradation data and subregion labels to construct input characteristics for the domain discriminator. Additionally, we design a feature extractor that produces label-related features, invariant across domains, effectively enhancing prediction precision. We evaluate our method using the publicly available C-MAPSS degradation dataset with a case study and ablation experiments.

Keywords
remaining useful life prediction, multisource domain adaptation, temporal conventional network, multilinear conditioning

1. Introduction

Prediction and health management (PHM) refers to a system's capability to accurately and promptly assess its present condition and the probability of malfunction within a designated time frame, providing decision recommendations for usage and maintenance activities. It is an essential component in the mechanical equipment industry, aimed at predicting and managing potential risks for the future, improving equipment safety and mission success, making sure that mechanical equipment operates safely and reliably, enhancing support efficiency, and reducing maintenance costs[1]. With the increasing development of the industrial era, enterprises have increasingly demanding requirements for the reliability of production systems, particularly critical components. Therefore, optimizing predictive health management is necessary and urgent.

As a critical component of PHM, RUL prediction aims to

(*) Corresponding author.
E-mail addresses: Y. Lv, lvyi913001@163.com, N. Zhou, zhouningxu2000@163.com, Z. Wen, 202121080938@std.uestc.edu.cn, Z. Shen, 2112204404@mail2.gdut.edu.cn, A. Chen, agchen@uestc.edu.cn,

model, analyze, and determine the timing of equipment failures through monitoring the operating state and studying the failure mechanism during equipment operation. It has broad application prospects in manufacturing, aerospace, and other fields[2]. In the PHM process, RUL prediction is the first step in identifying impending equipment failures. Subsequently, PHM proceeds to the second phase of health management, which involves scheduling rapid and accurate maintenance to minimize financial losses resulting from failures and enhance the safety and dependability of systems. Therefore, RUL prediction is crucial throughout the entire PHM process. However, in real-world scenarios, the accuracy of RUL prediction can be influenced by various factors, including environmental dynamics, data collection noise, and insufficient historical data[3]. To mitigate the influence of these factors and enhance the robustness of RUL prediction, domestic and international scholars have been integrating multiple cutting-edge technologies and continuously innovating RUL prediction methods. Consequently, RUL prediction has gained significant attention as a prominent research area in the field. Precise prediction of RUL for performance-degraded components has become an important issue in PHM.

Traditional RUL prediction methods mainly involve constructing degradation models to capture the dynamic degradation characteristics of products. This method requires analyzing the degradation mechanism of the product and combining prior knowledge to construct or select a probability model that is as consistent with the real degradation path as possible. Currently, commonly used degradation models include stochastic process models, general process models, and other process models[4]. In the early stages of research, computer power had not yet reached the level that it has today, and self-regressive algorithms[5], Markov models/chains[6], and support vector machines[7] were widely used. However, they cannot be widely developed because of the limitations of these methods and the challenging feature extraction required for researchers' knowledge reserves.

Compared with traditional models, deep neural networks have stronger data processing capabilities. Theoretically, deep neural networks are capable of approximating complex high-dimensional functions with great accuracy owing to their capability to approximate any continuous function with a high degree of precision. However, the challenges of gradient vanishing and explosion have limited the development of deep learning[8]. With the current optimizations of deep learning models, these issues have been significantly improved. Furthermore, the development of electronic component sensors and the exponentially increased computing power of computers have addressed the challenges of data acquisition, directly improving the predictive ability of mechanical system safety and reliability. With the abundance of collected signals, models can learn more accurately and comprehensively. Deep and complex models are crucial for achieving accurate RUL predictions. The availability of large-scale labeled data is also one of the key factors for the success of deep learning algorithms. Deep learning models, commonly referred to as data-driven approaches, possess the benefit of extracting features automatically. They are capable of extracting hidden features from the data, providing a more comprehensive representation of the degradation state at different operating stages, and establishing accurate mappings with labels. Data-driven methods can effectively extract discriminative features from intricate signals and directly perform RUL prediction tasks. For instance, Soda[9] employed an artificial neural network to predict the RUL of bearings. Khelif et al. [10] used support vector regression (SVR) models for engine RUL prediction. Aggab et al. [11] proposed an SVR and adaptive neuro-fuzzy inference system for lithium-ion battery RUL prediction. However, learning the functionality of these models directly from the raw signals is challenging because their structures are shallow.

On this basis, structurally more intricate deep neural networks have been further affirmed in the field of RUL prediction. Li et al.[12] studied a deep convolutional neural network (DCNN) model that had a longer training time than shallow networks did but achieved better results. Yang et al.[13] developed a dual-layer convolutional neural network (CNN) model that connected through intermediate reliability variables; the accuracy of RUL prediction using DCNNs can be enhanced by incorporating an additional convolutional layer. Lyu et al.[14] developed an LSTM-based RUL prediction method that, unlike traditional CNN models, used upper and lower bound estimation to increase the range of predictions. Deep neural networks are capable of handling complex systems.

Degradation data from complex systems often exhibit characteristics such as nonlinearity, hysteresis, and time-varying parameters, which make it difficult to achieve good results when using a single neural network for RUL prediction[15].

Therefore, in the past few years, an increasing number of studies have started to use hybrid neural networks for RUL prediction. These studies combine CNN with other methods such as long short-term memory (LSTM)[16], bidirectional long short-term memory(BiLSTM)[17], gated recurrent unit (GRU)[18], and extreme gradient boosting (XGBoost)[19] to improve prediction performance. The advantage of these hybrid models is that they can leverage the time-related attributes of the network to adjust to the attributes of intricate systems such as nonlinearity, hysteresis, and time-varying parameters. However, excessive stacking of models may lead to excessive model complexity, which can impact the effectiveness and prediction accuracy of the model. This situation means that new hybrid models need to be explored, which can organically combine various neural network structures to improve the accuracy and reliability of RUL prediction without being too complex and resulting in low time benefits. Thus, additional investigation is required to explore the synergistic effect between different network structures so that the role of hybrid models can be better explained.

The application of neural networks in the realm of RUL prediction has evolved from shallow to deep networks and then to hybrid networks, which has made feature extraction less challenging and facilitated the advancement of RUL prediction. However, despite the effective utilization of data, the issue of limited data availability in this domain remains unresolved. The model still has the problem of insufficient learning with the support of small amount of data. Limited data refers to the scarcity of extensive and accurately labeled degradation datasets, which poses challenges for comprehensive model training and validation. Data acquisition and measurement are difficult, and most open-source datasets have relatively small amounts of data. Various situations where no RUL label exists also occur in practical application scenarios. So we still need more diverse datasets to improve the robustness and generalizability of RUL prediction models. Furthermore, Transfer learning has good adaptability and can use similar domain data to make RUL prediction an important field. The cumbersome process of self-annotation can be reduced and the learning ability of the model can be enhanced by fully utilizing the previously annotated data and adapting to the domain through transfer learning. Zhang et al.[20]developed a transfer learning technique for estimating RUL using a BiLSTM recursive neural network, as described in their study, trained on a dataset that is associated with the target dataset, and finally fine-tuned it. Mao et al. [21] proposed a method for RUL prediction based on deep feature representation and transfer learning, and used transfer learning to assist in adjusting the characteristics of the bearing in question during the online phase.

The above-mentioned transfer techniques for RUL prediction are relatively traditional, and they all train on a single source domain before transfer. The degradation characteristics of the source domain are basically relatively single because of the small amount of data. With a high degree of similarity between the source and target domains, the transfer effect is relatively strong. However, in the actual wear and tear of performance degradation components, only the working conditions, sensors, and other factors can be used to determine whether the source and target domains exhibit a significant level of similarity, and their similarity cannot be determined. The expected results may not perform well in source domains with low similarity. Multisource domain transfer can solve the above problems well. Lately, some studies have been conducted on the application of multisource domain transfer methods, but they are still very rare. Shen et al.[22] proposed a transfer method using an intermediate domain, adding a new domain between the source and target domains for two transfers. Zou et al.[23] designed a multisource domain autoencoder model, optimizing the difference in dimension features during transfer. Lyu et al. [24] developed a multisource domain transfer RUL prediction model that improves feature differences.

With the existence of multisource domains, negative transfer may occur if only the overall difference in degradation features is minimized. In addition, with an increasing number of source domains, the data distribution of different domains differs, and the model's independent learning and optimization do not provide significant gains for RUL prediction compared to a single-source domain. In summary, the fundamental principle of the cross-condition RUL prediction proposed in this paper is

to use an efficient multilayer neural network architecture to improve prediction efficiency, combined with transfer learning based on multisource domains to provide massive data support for the network. In terms of optimization, the need to improve the accuracy of features extracted from multiple source domains and align source and target domains that exhibit greater similarity must be taken into consideration. In this context, taking labels into account becomes essential. This study introduces an unsupervised multisource domain adversarial network (UMDAN), which utilizes data from various operational conditions as source domains for acquiring diverse degradation knowledge and transferring it to the target domain to effectively perform the task of cross-domain RUL prediction. RUL prediction is crucial for predictive maintenance and operational efficiency, yet it is hindered by the scarcity of extensive and accurately labeled degradation datasets. These limitations pose challenges for comprehensive model training and validation. Our approach leverages multisource domain transfer learning to enhance the model's ability to generalize across different domains and operational conditions, thereby improving prediction accuracy and reliability. The main innovations and contributions of this paper are summarized as follows:

(1) Through efficient extraction of degradation features under multiple working conditions, the network will process data from multiple source domains in parallel. Then, the network undergoes additional optimization using adversarial learning techniques to enhance the model's efficiency.

(2) A mechanism for adapting features is developed to create domain-invariant features for each pair of domains. This mechanism aligns the features together.

(3) UMDAN improves the domain-invariant feature alignment through multilinear conditioning in the adversarial network unlike existing multisource domain networks. The specific method is to fully utilize the actual labels of the source domain while predicting the labels of the target domain. The labels and features are then multiplied, combined, and input into the adversarial network for domain discrimination. This approach effectively improves the accuracy of domain adversarial discrimination and enhances the transfer matching between multisource domains, accomplishing alignment between the source domains and the target domain without supervision.

(4) The proposed UMDAN model accomplishes prediction tasks across different conditions, enhancing the accuracy and adaptability of RUL prediction. We employ the publicly available NASA dataset CMAPSS to validate the proposed method.

The rest of this paper is organized as follows. Section 2 introduces the problem to be solved and the basic methods used. Section 3 provides a detailed description of the proposed network architecture and specific implementation methods. Section 4 describes the experimental process and result analysis. The final section summarizes the paper and proposes directions for future research.

2. Proposed method

2.1. Problem statement

First, a known dataset is used for the study. The dataset includes different failure modes and operating conditions. On the basis of this dataset, the necessary condition for transfer is that all operations are within the same label space domain[25]. The source domain can be transferred to the target domain if the set of operating conditions and failure modes is consistent. The source domain dataset can be represented by the formula

$$D_S = \{D_{S_j}\}_{j=1}^{M} = \{\{(x_{S_j}^i, y_{S_j}^i)\}_{i=1}^{n_{S_j}}\}_{j=1}^{M},$$

where $S_j$represents the j-th source domain. Moreover, the total set consists of M source domains, each of which is a subset representing different working conditions. The subset has n_{S_j} samples, $x_{S_j}^i$ depicting the i-th datum from the j-th source domain and $x_{S_j}^i$ representing the corresponding label of the j-th source domain for the i-th data sample. The target domain exhibits a comparable portrayal to that of the source domain, i.e., $D_T = \{x_T^i\}_{i=1}^{n_T}$. However, the absence of RUL label y in the target domain is attributable to the fact that the model is tasked with predicting said RUL label. Unlike the source domain, only one target domain exists, which has n_T samples, and x_T^i represents the i-th sample. The dimension of each sample is determined by the sliding window. For the initial sample, given a time step t, we define function

$\varphi_{T_w}, i.e. \varphi_{T_w}(x^i) = \{(x_{t-T_w}^i, \dots, x_{t-1}^i)\}_{t=T_w+1}^{T_i}$, which separates each sequence with a size of T_i into consecutive time intervals

of specific magnitude T_w. Therefore, a single-source domain sample is represented as $x_{S_j}^i = \varphi_{T_w}(x_{Source_j}^k)$, the data from the source domain is depicted as

$D_S = \{\{(\varphi_{T_w}(x_{Source_j}^i), y_{S_j}^i)\}_{i=1}^{n_{S_j}}\}_{j=1}^{M}$, and the label is the true label. The data from the target domain are depicted as $D_T = \{\varphi_{T_w}(x_{Target_j}^i)\}_{i=1}^{n_T}$. The transition from the source domain to the target domain solely employs the adversarial learning technique, which may lead to the misalignment of adjacent features and labels, resulting in a reverse effect of the transfer learning. In an environment with multisource domains, an increase in the amount of data and the occurrence of this situation in each source domain-to-target domain transfer will cause a sharp increase in model error, resulting in poor overall performance. A feature extractor is used to extract data from both the source domain and the target domain. Subsequently, the extracted features are inputted into the DANN network for adversarial learning.

As illustrated in Figure 1(a), the original aim of transfer learning is to synchronize the features of the source domain and target domain with the degradation process of RUL. However, the extraction of deep features is lacking because of insufficient degradation feature extraction in the underlying network of DANN. Therefore, in the adaptive process, mapping errors may occur, leading to incorrect transfer and resulting in negative transfer effects. As illustrated in Figure 1(b), the correct adaptive transfer should be m1, but the negative transfer phenomenon of m2 occurs due to the misalignment between the features and the RUL label.

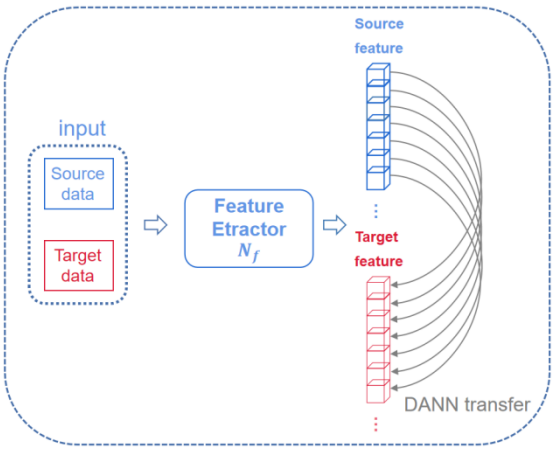


Fig.1(a) Schematic diagram of the domain match.

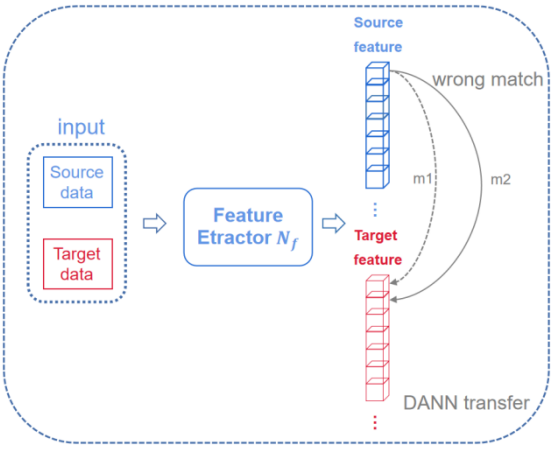


Fig.1(b) Schematic diagram of the domain mismatch.

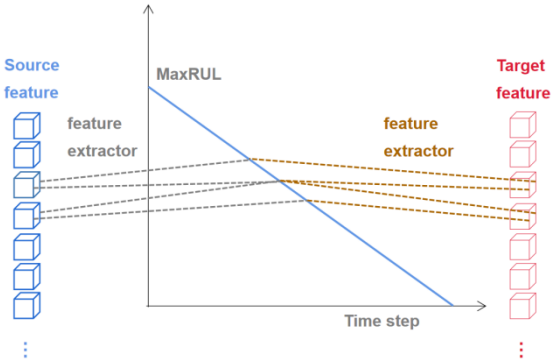


Fig.2(a) Detail diagram of the domain match.

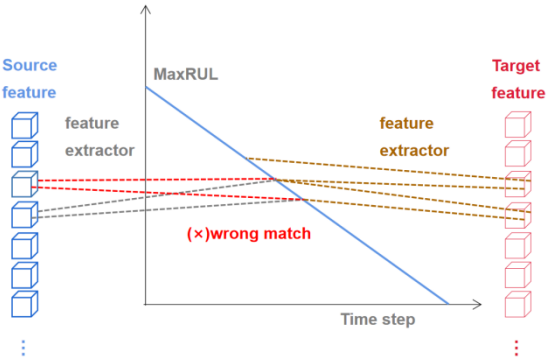


Fig.2(b) Detail diagram of the domain mismatch.

According to the actual time period and its corresponding features, this mismatch phenomenon occurred because the features extracted from adjacent time periods are very similar, which greatly increases the likelihood of mismatch and consequently results in errors. As shown in Figure 2, the similarity of features in adjacent time steps leads to a non-unique mapping relationship with the labels, forming a situation of multiple mappings. This condition is one of the causes of the mismatch phenomenon in deteriorating feature RUL labels.

On the Figure 2(a) is a normal match and on the right is a mismatch, which fully demonstrate the mismatch phenomenon of the deteriorating feature RUL label. This issue is solved by considering the relationship between labels and features: the source domain has true labels, and the more source

domains there are, the more true labels there are. However, at the same time, the more source domains there are, the greater the impact of mismatch. Therefore, the focus is on the labels and features, where their true correlation can greatly improve the problem of mismatch. This paper uses multisource domains to solve the problem of learning a single problem and to address the impact of mismatch in multisource domains by combining labels and features.

2.2. Main body of the network

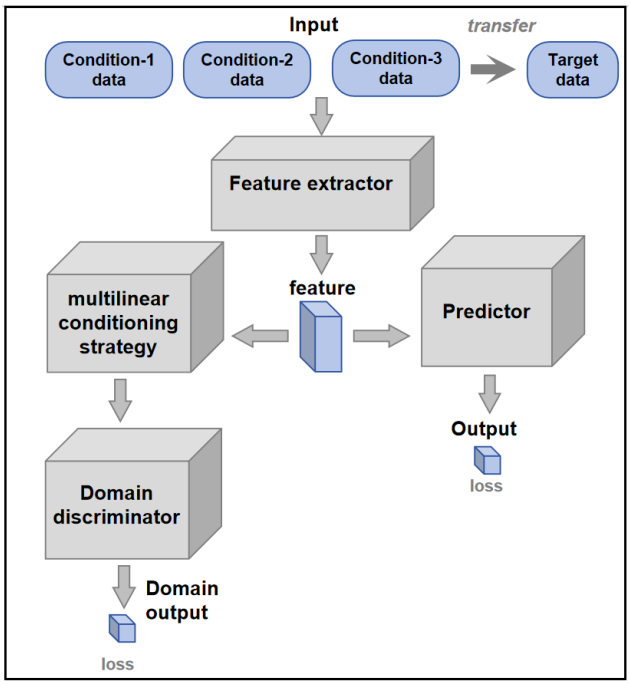


Fig.3 The overall architecture of the network.

As shown in Figure 3, the overall network is divided into four components: feature generation network, RUL prediction network, subdomain classification network, and domain discrimination network.

An accurate and efficient RUL prediction network is highly correlated with the main part of the network. On top of the main network, further optimizations can be made locally. Therefore, choosing a network that fits the current conditions is crucial. First, multisource domain RUL prediction needs to address several issues. The primary concern is that data sourced from multiple domains may have different time steps, as these data are collected under different operating conditions, leading to variations in failure modes. Therefore, using traditional RNN alone would require complex processing, adding to the intricacy of the task. In contrast, a temporal convolution network (TCN) can effectively handle data with different time steps. In addition, multisource domain RUL prediction involves data spanning across different domains. TCN can adaptively learn the feature relationships across domains, capturing trends and patterns in the data. Therefore, when performing multisource domain RUL prediction, TCN can handle data from multisource domains, improving the overall prediction accuracy of the model. Lastly, the inherent connections within multisource domain data are complex, with closely related features and labels, and cross-relationships exist between different labels and features. TCN deep learning can discover these inherent connections and enhance the model's generalization capability.

TCN is a structural innovation built upon CNN[26]. CNN is constrained by the size of the convolutional kernel, where a small kernel lacks the ability to capture the global context, while a large kernel leads to model complexity. Therefore, TCN emerged as a crucial solution. It combines the advantages of multilayered convolution in CNN and the ability of RNN to capture long and short-term dependencies while avoiding problems such as gradient explosion and poor long-term memory[27]. In previous sequence modeling tasks, LSTM and GRU, among other RNN, were often employed[28]. Nevertheless, in the area of RUL prediction, with the limitations of the dataset and strong temporal dependencies, TCN demonstrates superior performance. TCN preserves more extended memory than LSTM can, has a reinforcing impact on RUL prediction, and achieves better results in sequence modeling tasks[29].

TCN utilizes a 1D fully convolutional network architecture. The size of each hidden layer is identical to the size of the input layer, and zero padding is applied to ensure consistent dimensions across subsequent layers. Nevertheless, during prediction, we want the model to only utilize past information to predict future remaining life span and not use future information to influence past predictions. Therefore, causal convolution is used to prevent any information leakage from future time steps to past time steps to generate the output at time t[30]. As illustrated in Figure 4, the input at time t and the input from a previous layer at an earlier time are used. Each hidden layer has the same length as the input layer, and zero padding is applied to ensure equal length in subsequent layers.

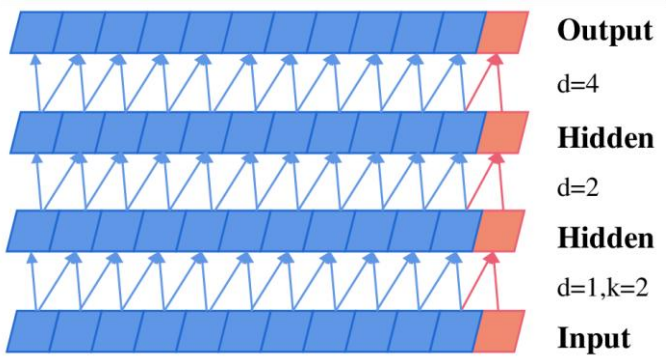


Fig.4. Schematic of causal convolution.

The causal constraint implies that the TCN builds a mapping relationship for the input time step T sequence $(x_0, x_1 \ldots, x_{T-1}, x_T)$, such that $y_T = f(x_0, x_1 \ldots, x_{T-1}, x_T)$ and eventually obtains $(y_0, y_1 \ldots, y_{T-1}, y_T)$. During prediction, using data beyond time step t to predict the result at time step t is not allowed. In other words, $y_t = f(x_0, x_1 \ldots, x_{t-1}, x_t)$ cannot depend on data $(x_{t+1}, x_{t+2} \ldots, x_{T-1}, x_T)$ beyond time step t to predict the result at the current time step.

This condition also matches the real situation in bearing RUL prediction. Unlike in machine translation, complete sequential data can be used for RUL prediction. Bearing degradation data are generated according to the time axis in the forward direction. The future data cannot be obtained in the current prediction; otherwise, RUL prediction would obviously lose its meaning. This setting is more restrictive than that of the seq2seq model, but it is more in line with actual prediction.

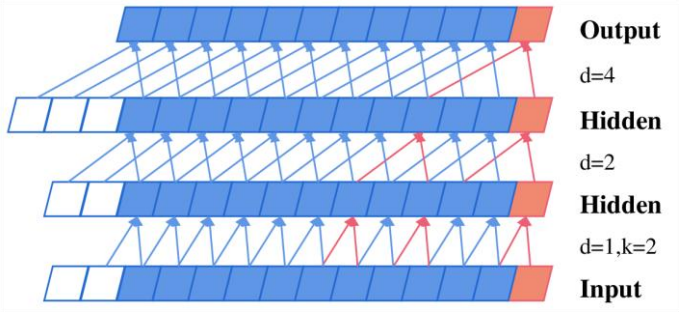


Fig.5. Schematic of dilated causal convolution.

Another issue occurs when reviewing consecutive time steps in the network, that is, only the total count of layers in the network can be reviewed. This issue is addressed by using dilated convolutions[31] to acquire input from every d steps away from t: $(x_{t-(k-1)d} \ldots, x_{t-2d}, x_{t-d}, x_t)$, where k represents the size of the kernel. Dilated convolutions allow the network to observe (k-1)d previous time steps, expanding the perceptive area of every layer $d = O(2^i)$. By introducing dilation parameters, the convolution kernel can skip some data points in the input data to obtain a longer effective history, reducing the number of internal parameters while ensuring effective history, thus reducing the computational power and storage requirements of the model[32]. This approach greatly enhances the efficiency in the RUL prediction field. As shown in Fig. 5, one can view all the inputs reaching the bottom along the red line from top to bottom, which means that the prediction of the output (as an example at time T) uses all the inputs from the effective historical data.

TCNs are designed to handle sequential data by capturing long-range dependencies with convolutional layers, which makes them suitable for time-series data like degradation signals. However, TCNs alone may not handle domain discrepancies well when the training and test data come from different operational conditions.

2.3. Transfer learning network implementation

Domain adaptation is the transfer learning approach in which the data distribution varies between the target and source domains, but the objective remains unchanged[33]. The objective of domain adaptation, also known as domain adaptation, is to adapt and reduce the differences in data distribution within the same feature space in a predefined source and target domain environment. Domain adaptation initially made breakthroughs in image classification tasks and has been widely applied in predictive domains as deep learning techniques continue to advance. Many domain adaptation techniques have achieved excellent results in different types of predictive tasks. In time series prediction tasks, domain adaptation techniques LSTM are widely employed[34]. The domain adversarial adaptation network (DAAN) is among the most commonly utilized domain adaptation approaches[35]. As shown in Figure 6, the proposed DAAN implementation is made up of four parts. The feature generation network N_f is used to produce cross-domain invariant features, the domain discriminator network N_d is utilized to differentiate whether the features come from the source domain or the target domain, the life prediction network N_p is used to predict the target function, and the subdomain classification network N_c is used to classify subdomains. The DAAN component is associated with N_f, N_d, and N_c. DAAN utilizes a domain discriminator to differentiate

whether the features from the feature generation network originate from the source domain or the target domain while simultaneously allowing for multilinear adjustments from the subdomain classification network. Adversarial learning, while powerful for aligning feature distributions between source and target domains, can have potential negative effects. One primary concern is the risk of negative transfer, which occurs when the alignment process misaligns adjacent features and labels. This misalignment can lead to incorrect feature-label associations, ultimately degrading the model's performance. For example, if the adversarial network forces features from different operational conditions to align too aggressively, it might overlook subtle yet critical differences between these conditions, resulting in poor generalization on the target domain. References that discuss similar challenges include Ganin and Lempitsky[36], and Tzeng et al. [37], which highlight the delicate balance required in adversarial training to avoid such pitfalls.

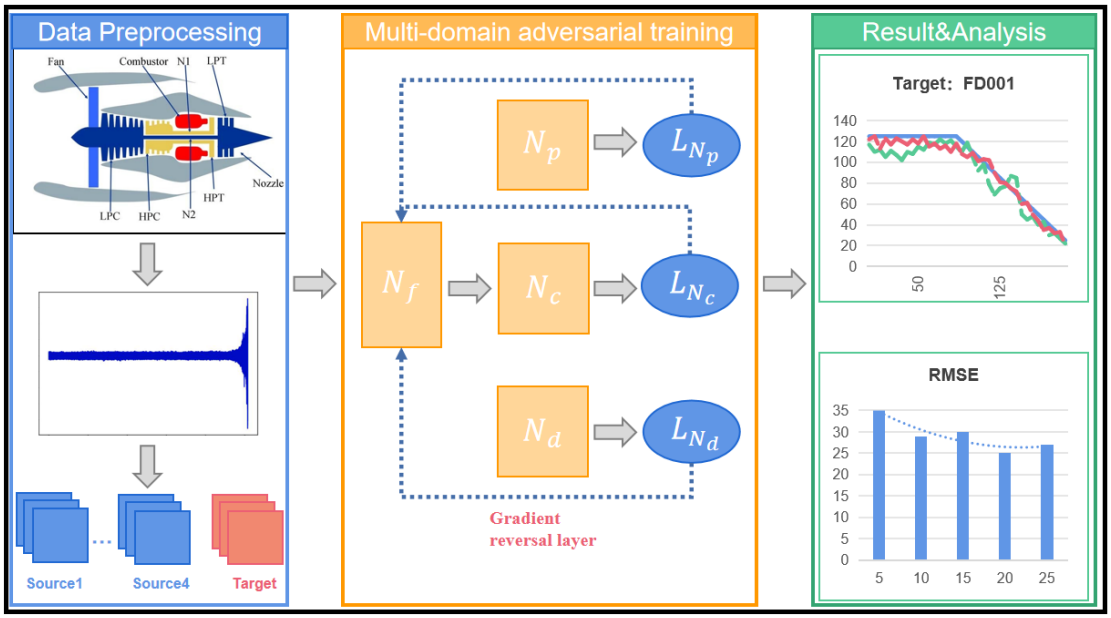


Fig.6. Domain adversarial adaptive network structure diagram.

The most important part of the DAAN network is the gradient reversal layer (GRL), which adopts the idea of adversarial learning and automatically reverses the gradient direction during backpropagation. During forward propagation, the GRL does not impact the input of the feature, acting mainly on backpropagation. During backpropagation, the gradient forwarded from the domain discriminator is multiplied by a factor and transformed to its negative value using GRL. Specifically, as shown in (2.1), both the domain classifier and RUL predictor take input from the feature extractor, but the domain classifier aims to maximize the domain classification loss by confusing the target domain data with the source domain data, while the RUL predictor aims to minimize the RUL prediction loss, and both losses are balanced through adversarial training[38].

$$\frac{\partial L_{N_d}}{\partial \theta_{N_d}} \xrightarrow{GRL} -\lambda \frac{\partial L_{N_d}}{\partial \theta_{N_f}} \tag{2.1}$$

In this case, the loss function L_{N_d} is reversed after passing through the GRL layer to optimize the parameters. The optimization process for the parameters is opposite: the domain discriminator network aims to minimize L_{N_d}, while after GRL, the optimization direction for the gradient of the feature generation network is towards increasing its value L_{N_d}. This establishes a hostile connection between the two, and as a result of training, the feature generator has the ability to extract domain-invariant features. Furthermore, the discriminator is unable to differentiate the origin domain of the input features [39].

At the same time, the GRL parameter λ should dynamically change with the progress of the experiment. γ is a constant of 10, and p is a dynamically changing indicator parameter expressing the proportion of the current iteration count to the total iteration count, as shown in Equation (2.2).

In the DANN architecture, in order to help the model adapt to new data distributions faster and reach the optimal solution quickly, the learning rate μ also dynamically adapts the results,

as shown in Equation (2.3), where μ_0 is the initial learning rate, and α and β are defined as hyperparameters. Therefore, the objective function of DANN is a max-min function. L_{N_d} refers to the domain discriminator's loss, which needs to be minimized in Equation (2.4), and L_{UMDAN} is the overall network loss. In Equation (2.5), the overall loss L_{UMDAN} needs to be minimized. Therefore, the reversed L_{N_d} needs to be maximized.

$$\lambda = \frac{2}{1 + e^{(-\gamma \cdot p)}} - 1 \tag{2.2}$$

$$\mu = \frac{\mu_0}{(1 + \alpha \cdot p)^{\beta}} \tag{2.3}$$

$$L_{N_d}(\theta_{N_d}) = \sum_{j=1}^{M} \left(E_{x^i_{Source}} log(N_f(\varphi_{T_w}(x^i_{Source_j})) + E_{x_{Target}} log(N_f(\varphi_{T_w}(x_{Target}))) \right) \tag{2.4}$$

$$\begin{aligned} &L_{UMDAN}(\theta_{N_f}, \theta_{N_d}, \theta_{N_p}, \theta_{N_c}) \\ &= \sum_{j=1}^{M} \min_{\theta_{N_f}, \theta_{N_d}, \theta_{N_p}, \theta_{N_c}} \alpha E_{(x^i_{Source_j}, y^i_{Source_j}) \in SD} L_{N_p}(N_p(N_f(\varphi_{T_w}(x^i_{Source_j}))), y^i_{Source_j}) \\ &+ \beta E_{(x^i_{Source}, g^i_s) \in SD} L_{N_c}(N_c(N_f(\varphi_{T_w}(x^i_{Source}))), g^i_s) - \gamma L_{N_d}(\theta_{N_d}) \end{aligned} \tag{2.5}$$

DANNs aim to learn domain-invariant features by using adversarial training to reduce the discrepancy between source and target domain feature distributions. This approach is effective in transfer learning scenarios but might struggle with time-series data's temporal dependencies.

3. Unsupervised multiple sub-domain adversarial network with temporal convolutional network

If trained with several source domains, then the variances among them could be further magnified. If the feature extraction component is designed to be simple or shallow, then it may not capture the differential features pertaining to the data from the source domain adequately, resulting in a domain mismatch among the source and target domains. In addition to incorporating the domain classifier and domain adversarial loss strategy in the DANN network, considering the introduction of a label alignment strategy can enhance the effect of domain adaptation. Therefore, this article suggests a network for adversarial adaptation with multiple source and subdomains. Our method integrates TCN with DANN within a multisource domain transfer learning framework. This combination allows us to leverage TCN's ability to capture temporal dependencies and DANN's strength in reducing domain discrepancies. Additionally, the use of multiple source domains helps to further enhance the model's generalizability and robustness.

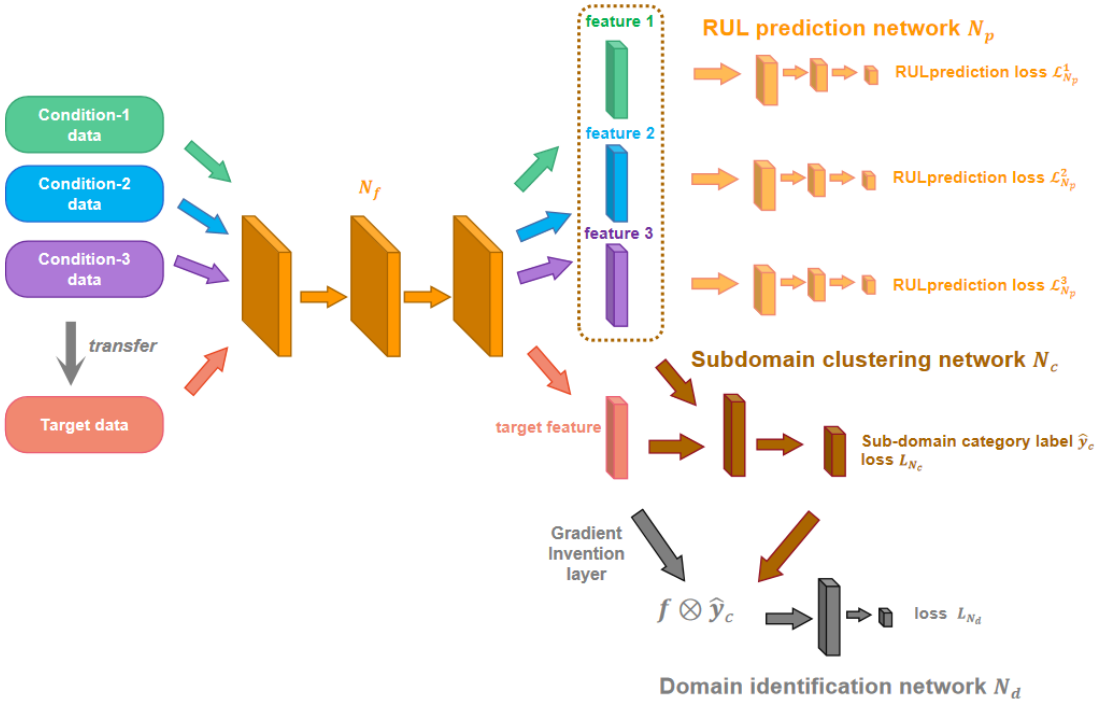


Fig.7. The network structure diagram of the UMDAN.

The overall framework is illustrated in Figure 7. The acquired cross-condition dataset is built on the publicly available C-MAPSS dataset. Based on the requirements of cross-condition and the number of source domains, the dataset is divided into three different condition source domain data and one target domain data. As the main framework for cross-condition transfer tasks, it is constructed by TCN, which is proficient in handling time-series data. The objective of this network is to achieve adaptive transfer from multisource domains to the target domain. The multisource domain data are divided into three source domains. Furthermore, the information from each source domain is optimized separately. The overall network is divided into four components: feature generation network, RUL prediction network, subdomain classification network, and domain discrimination network. First, the feature extraction network, which is the main TCN framework, takes all the data as inputs and serves as the common part of the entire framework, playing a crucial role. The RUL prediction network serves as the central component of the framework, and the network's performance is ultimately judged based on its output. Subdomain classification network is used to divide the subdomains and align labels with features, and its output is combined linearly with the features of the input domain discrimination network instead of being directly outputted. The domain discrimination network incorporates an adversarial mechanism for adaptive transfer of data. A method using the subdomain classification network is proposed to reduce the training error of multisource domain data. This method optimizes the domain discrimination network to more accurately discriminate among source and target domain data in DANN, thereby reducing the occurrence of mismatch phenomena. The multisource domain network designed in this paper proposes a new type of DANN network when most existing multisource domain frameworks fail to address the mismatch error issue. The innovative part of the multisource domain network lies in the DANN part. In the traditional framework that adapts transfer through the back-propagation layer, if only features are used for domain discrimination, then it may be limited by the performance of the feature extraction network, resulting in shallow features and lack of a unique mapping relationship. Similar features between adjacent times may cause cross-mismatch or multiple mappings from features to labels. This situation is more severe in the multisource domain field with massive data, ultimately leading to increased transfer errors and poor predictive performance of the model.

Therefore, the problem of overfitting can be discovered easily when referring to previous supervised learning methods, thus increasing the phenomenon of domain mismatch and elevating the proficiency of multisource domain models. Unsupervised learning networks can effectively solve the problem of overfitting[40]. Based on unsupervised learning, an unsupervised multisource domain subdomain adversarial network is designed, where optimizing the DANN network is a key part. The primary method is to optimize the input of the adversarial transfer part. Considering that a corresponding relationship exists between individual subdomains and features after subdomain partitioning, they are associated. This condition allows the domain discriminator to better identify the unique information in the domain when discriminating the features associated with the subdomain, reducing the transfer error of individual time steps. The following is a specific implementation.

The idea of subdomain partitioning network is to initially separate the label space into RUL subdomain regions, where every subdomain region represents to a category. The entire process of device life span is divided into several substages, each representing a different degradation state, and then the corresponding features for each substage are established. The specific partitioning strategy is designed based on data features and application scenarios as follows (3.1). A crucial step is to observe that the choice of stages and the number of subdomains, especially the number of subdomains in the partitioning, should be reasonable.

$$Y_c = unique(\{y_c^i = rounddown(\frac{RUL_{max} - t}{RUL_{max}}) * n_c | 0 \le t \le RUL_{max}\}) \quad (3.1)$$

The unique (·) function is utilized to eliminate duplicate values and merge duplicate subdomain partitions into a single category. RUL_{max} represents the highest amount of remaining useful life, which is a fixed value. rounddown(·) is a floor function, and using floor rounding for RUL prediction is more consistent with preventative strategies before component failure.

The source domain has RUL labels; hence, subdomain partitioning can be performed directly. However, because the target domain does not have labels, partitioning needs to be performed through the network. The subdomain classification network is built using the features generated by the feature generation network as input. The network uses cross-entropy loss function to calculate the error in subdomain classification by using the following formula:

$$L_{N_c} = \sum_{j=1}^{x} \sum_{i=1}^{n_{Source_j}} \frac{1}{n_{Source_j}} (-y^i_{Source_j c} log \hat{y}^i_{Source_j c} - (1 - y^i_{Source_j c}) log(1 - \hat{y}^i_{Source_j c})) \quad (3.2)$$

where n_{Source_j} denotes the overall count of training samples in the j-th source domain, and x denotes the overall count of source domains. This study considers both dual-source domain and tri-source domain settings. $y^i_{Source_j c}$ denotes the accurate subdomain classification label of the i-th sample in the j-th source domain, while $\hat{y}^i_{Source_j c}$ represents the predicted subdomain label. The parameters and weights of the subdomain classification network are optimized and updated using the backpropagation technique. The pre-trained subdomain classification model takes the target degradation characteristics produced by N_f as input, yielding pseudo-labels $\hat{Y}_c^T$ for subdomain categories.

Then, the improvement of DANN is targeted, with the core idea of associating subdomains with features. Multilinear adjustment is used instead of directly connecting features to the obtained pseudo-subdomain labels. The multilinear conditioning strategy enhances the model's ability to learn complex relationships between features and labels across different domains. By integrating features and labels through multilinear transformations, this strategy captures higher-order interactions that are crucial for accurate RUL predictions. This approach ensures that the model can leverage information from multiple source domains more effectively, leading to improved prediction accuracy and efficiency. Multilinear adjustment can effectively model the relationships between multiple features and subdomains, capturing complex interactions between features. It can also better handle the complex relationships between multi-sensor data and labels. Multisource domains mean different sensor models and data collection methods, leading to different data storage formats and possibly different dimensions. However, for similar degrading components, their intrinsic nonlinear relationships are more difficult to build than linear relationships. The intrinsic connection between labels and multi-sensor data needs to be established by constructing a multilinear architecture. This way, the diverse structure underlying the intricate data distribution and the interactive multiplication between features and classifier predictions can be grasped[41]. The computation of the adjusted domain discriminator network input features is as follows: (3.3) represents the computation for the source domain, and (3.4) represents the computation for the target domain. The labels linked with the source domain and target domain must be dissimilar. The source domain has true labels, so the true labels are directly used for partitioning. $y_{c(a)}^{Source_j}$ represents true labels belonging to category "a" in the j-th source domain. From another perspective, the target domain does not have true labels, so pseudo-labels are used. $\hat{y}_{c(a)}^{Target}$ represents the pseudo-subdomain label for category "a" in the target domain.

$$\mathrm{H}_{\mathrm{i}}^{Source_j} = N_f(\varphi_{T_w}(x^i_{Source_j}) \otimes y_{c(a)}^{Source_j} \quad (3.3)$$

$$\mathrm{H}_{\mathrm{i}}^{\mathrm{Target}} = N_f(\varphi_{T_w}(x^i_{Target}) \otimes \hat{y}_c^{Target} \quad (3.4)$$

The training set utilized in the RUL prediction network, located at the network's core, consists of data from the source domain. In contrast, data from the target domain are employed as the test set. To begin with, RUL labels are available for the source domain, which are subsequently utilized to predict pseudo-RUL labels for the source domain data. The pseudo-labels are continuously updated to iterate and optimize the network. The network uses the mean absolute error function to calculate the regression error, as shown in formula (3.5).

$$L_{N_p} = \sum_{j=1}^{x} \sum_{i=1}^{n_{Source_j}} \frac{1}{n_{Source_j}} |N_p(N_f(\varphi_{T_w}(x^i_{Source_j}))) - y^i_{Source_j}| \quad (3.5)$$

Within this given context, the number of training samples in the j-th source domain, represented by n_{Source_j}, and the true

RUL label of the i-th sample, represented by $y^i_{Source_j}$, are of significance. The backpropagation technique is employed to optimize and update the parameters and weights of the fault classification network.

The classification error of the domain discriminator network is calculated by utilizing the binary cross-entropy loss function to distinguish between the new hidden features that have been subjected to multilinear adjustments in the source and target domains. An exact mapping function relationship between the deteriorated features and RUL labels is achieved by using the root mean square loss function to quantify the prediction error of the label prediction network.

$$L_{N_d} = \sum_{j=1}^{x} -d^i log N_d(H_i) - (1-d^i) log(1 - N_d(H_i)) \quad (3.6)$$

In this context, d^i represents its true class label,indicating whether the i-th new hidden feature belongs to the source or target domain. H_i comprises features that are a multilinear combination of j source domains and the target domain. $N_d(H_i)$ ranges from 0 to 1, representing the origin of $N_d(H_i)$ belonging to either the source or target domain. The discriminator L_{N_d} is optimized by minimizing its loss. At the feature extractor level, the overall loss is minimized through a GRL, which maximizes L_{N_d} and confuses N_d, enabling adaptive adversarial transfer. Represented by the following formula is the method of adversarial domain adaptation, which captures min-max game:

$$\begin{gathered} \min_{N_d} L_{N_d} \\ \min_{N_f} L_{N_c} + L_{N_p} - L_{N_d}\,(\mathrm{GRL}) \\ \max_{N_f} L_{N_d} \end{gathered} \quad (3.7)$$

The total objective function of the UMDAN network is composed of three components: sub-domain classification objective function, domain discrimination objective function, and RUL prediction objective function. It is represented as follows in the equation:

$$\begin{aligned} L_{USDAN}(\theta_{N_f}, \theta_{N_p}, \theta_{N_d}, \theta_{N_c}) \\ = \alpha L_{N_p}(\theta_{N_p}) + \beta L_{N_c}(\theta_{N_c}) \\ - \gamma L_{N_d}(\theta_{N_d}) \end{aligned} \quad (3.8)$$

Whereas, $\theta_{N_f}, \theta_{N_p}, \theta_{N_d}, \theta_{N_c}$ represents the parameters of the four networks in UMDAN. By applying the formula below, the tradeoff parameter γ smoothly shifts from 0 to 1. Here, p represents the linearly changing training progress ranging from 0 to 1, ε is a constant set to 10, and The current iteration number and the total number of iterations are indicated by iteration and miteration respectively.

$$\beta = 2/(1 + e^{-\varepsilon \frac{iteration}{miteration}}) - 1 \quad (3.9)$$

$$\gamma = \frac{2}{1 + exp(-\varepsilon p)} - 1 \in [0,1) \quad (3.10)$$

To optimize the weight parameters of the UMDAN model, we employ the stochastic gradient descent algorithm. The direction of gradient optimization is determined during the backpropagation process. The following formula outlines the model parameter update, where the learning rate is represented by δ.

$$\begin{cases} \theta_{N_f} \leftarrow \theta_{N_f} - \delta \dfrac{\partial L_{N_p}(\theta_{N_p}) + \mu L_{N_c}(\theta_{N_c}) - \gamma L_{N_d}(\theta_{N_d})}{\partial \theta_{N_f}} \\ \theta_{N_p} \leftarrow \theta_{N_p} - \delta \dfrac{\partial L_{N_p}}{\partial \theta_{N_p}} \\ \theta_{N_d} \leftarrow \theta_{N_d} - \delta \dfrac{\partial L_{N_d}}{\partial \theta_{N_d}} \\ \theta_{N_c} \leftarrow \theta_{N_c} - \delta \dfrac{\partial L_{N_c}}{\partial \theta_{N_c}} \end{cases} \quad (3.11)$$

4. Experiment

4.1. Dataset Description

Table I. DESCRIPTION OF C-MAPSS DATASET.

Subset	FD001	FD002	FD003	FD004
No. Of Training Engines	100	260	100	249
No. Of Test Engines	100	259	100	248
Operating Conditions	1	6	1	6
Fault Modes	1	1	2	2

The commonly used C-MAPSS dataset was employed to demonstrate the feasibility of the approach[42]. The C-MAPSS dataset, which stands for Commercial Modular Aero-Propulsion System Simulation, is a widely used benchmark dataset for RUL prediction. It contains run-to-failure data for turbofan engine units under various operational conditions and fault modes, making it a suitable choice for validating our multisource domain transfer learning approach. The advantage of the C-MAPSS dataset lies in its public availability, which has facilitated numerous experiments in this domain. It simulates the entire life cycle of aircraft engines and is divided into four subsets, facilitating the implementation of multisource domain transfer methods. Table I presents the fundamental details of the dataset, with each subset having different operating and fault conditions. For example, there are 100 training engines and 100

test engines in subsets FD001 and FD003, with few operating conditions and failure modes. The number is relatively small; therefore, these two subsets can be considered similar. Subsets FD002 and FD004 are more intricate than the previous subsets, with a larger number of training and test engines and six different operating conditions. Moreover, the maximum life cycles of the engines in the dataset vary. The training dataset includes the corresponding RUL values for each cycle, while the test dataset introduces random stoppages during the cycles. These datasets consist of rows representing samples, with 26 columns for each row. The first column is engine ID, the second column represents the current cycle, and the third to fifth columns refer to the operating conditions. The remaining 21 sensors' values are stored in columns 6 to 26, but not all of them provide RUL prediction information. Among the sensors, S1, S5, S6, S10, S16, S18, and S19 maintain a relatively consistent value during the engine's lifetime[43]. These constant values provide no positive feedback for training and may cause prediction errors. Therefore, the remaining columns, excluding the constant data, are extracted as training data. There are a total of 14 sensors with changing values, which are then labeled and normalized using min-max normalization as indicated in Equation 4.1. This step ensures that all features contribute equally to the model training process and improves convergence speed.

$$x_i' = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)} \tag{4.1}$$

where $max(x_i)$ denotes the maximum value of the feature signal x_i in data sample x, and $min(x_i)$ symbolizes the minimum value. Therefore, the final normalized data range is $x_i' \in [0, 1]$.

Determining the anticipated output value of the input data for RUL prediction problem is a challenging task. Assessing the precise health condition and estimating the RUL of a system at each time step in numerous industrial applications is often impractical because of the absence of an accurate physics-based model[44]. For the experimental setup dataset, segmented linear degradation models have been validated as appropriate and effective[45]. The segmented linear model divides the operating period of the working unit, with a focus on distinguishing the operational phases of the degrading unit. According to this theory, the operating process is divided into a constant operating period and a linear degradation period. In the early stages, no failures are assumed to occur, and the RUL value remains constant at its maximum value. After a certain time, linear degradation occurs until a failure event takes place. Furthermore, the maximum RUL value cannot be arbitrarily set, as it affects the prediction performance[46]. On the basis of extensive experiments using this dataset, the appropriate maximum RUL value is determined and set to 125. For each time cycle of the dataset, the corresponding label y_i for the RUL can be obtained by using formula 4.2. Then, a technique known as the sliding time window is utilized to magnify the feature data for each time period, as shown in formula 4.3. This method helps the model to learn from both short-term and long-term degradation patterns.

$$y_i = \begin{cases} RUL_{max} & y_{RUL} \geq RUL_{max} \\ y_{RUL} & y_{RUL} < RUL_{max} \end{cases} \tag{4.2}$$

$$\varphi_{T_w}, i.e. \varphi_{T_w}(x^i) = \left\{(x^i_{t-T_w}, \ldots, x^i_{t-1})\right\}_{t=T_w+1}^{T_i} \tag{4.3}$$

4.2. Parameter Setting

The setting of hyperparameters will be determined using grid search, which exhaustively searches within the given range of hyperparameters, records performance data, and ultimately selects the best combination[47]. The experimental setup includes a comparison section, where multiple methods will be used for performance comparison. Equivalent parameters will be used to make the comparative experiments effective and reliable.

Figure 8 demonstrates this concept. The convolution kernel size is 3 with a count of 5 in the TCN model. TCN's characteristic is to maintain consistent input and output, which is set to 40 in this experiment. In the TCN model, causal convolution is used with zero-padding on the left side to ensure that no future information is relied upon. At the same time, when the convolution kernel slides to the first position of the sequence, it can consider the zero values on both sides of the input sequence, ensuring that the length of the convolution output is consistent with the input sequence length. As shown in Equation 4.4, the size is 3, $n_{kernel\ size}$ represents the convolution kernel, $n^i_{padding\ zero}$ represents the number of left padding zeros, x represents the x-th hidden layer, and l represents the dilation rate.

$$n^x_{padding\ zero} = l_x \times (n_{kernel\ size} - 1) \tag{4.4}$$

TCN is a sequence modeling method based on CNN, which

is commonly used for handling sequence data such as time series. It utilizes multiple convolutional layers and repeated residual blocks to effectively capture long-term dependencies in the sequences. However, in the TCN model, the convolutional layers have a large number of parameters, which can lead to overfitting if proper regularization methods are not applied. The risk of overfitting is mitigated by adding dropout layers after the TCN model, which randomly set a portion of the neuron outputs to 0, reducing the cooperation between neurons and enhancing the robustness and adaptability of the model.

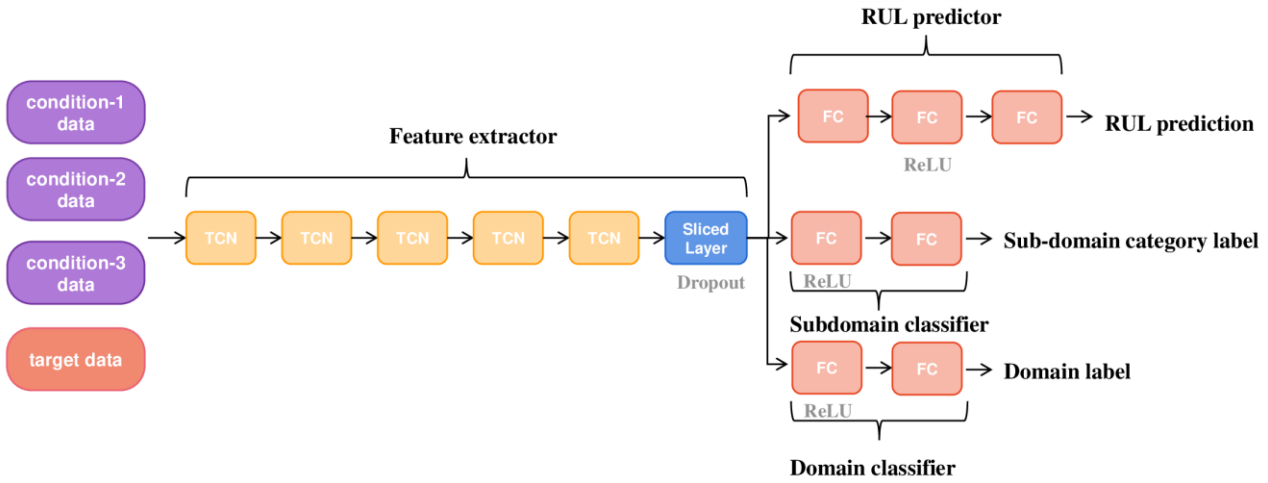


Fig.8 Detailed network structure of UMDAN

The network that is utilized to predict the RUL is made up of three linear layers, along with a dropout layer, and a ReLU activation layer is strategically placed after the first two linear layers to enable nonlinear transformations. This approach increases the flexibility and expressive power of the model, improves its capability to model complex features, and enhances its prediction performance. Moreover, the activation sparsity characteristic helps diminish overfitting and enhance the model's capacity to generalize. The predicted outcome is the output of the last fully connected layer. The subdomain classification network has almost no difference in structure compared with the RUL prediction network, except for one less fully connected layer and the output dimension of the last fully connected layer changed to the number of subdomains. A new hidden feature is the input of the DANN portion, which comprises multilinear conditions. The size of its input dimension is determined by multiplying the linear expansion dimension of the degradation feature generated by the algorithm with the number of subdomains.

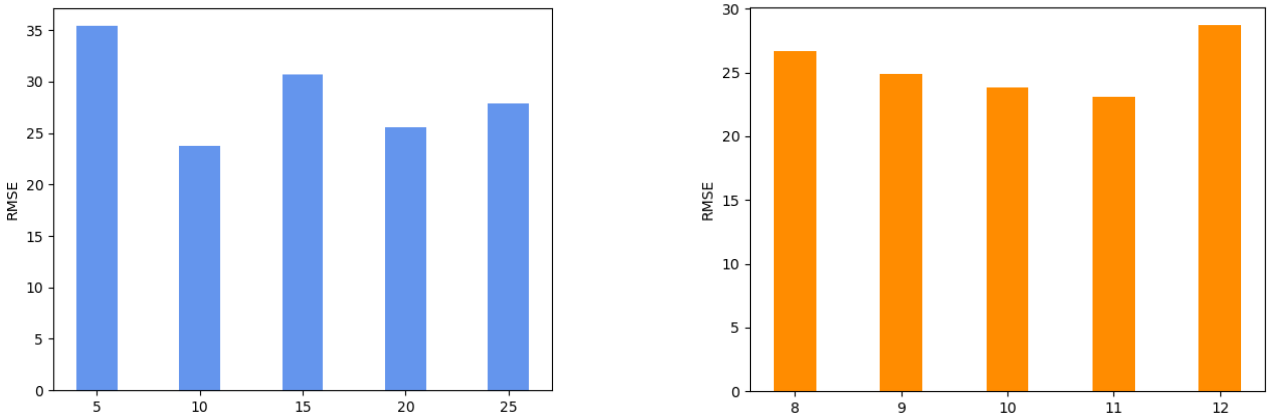
The detailed parameter settings are outlined in Table II. We selected a convolutional kernel size of 3 to enhance convergence due to the relatively small dataset. This smaller kernel size accelerates model convergence and training while effectively capturing short-term anomalies. For the C-MAPSS dataset, the batch size is set to 32. Although a larger batch size could increase memory consumption, choosing 32 strikes a balance between training speed and memory efficiency. We adopted a learning rate of 0.0003, which is small enough to ensure smooth convergence and mitigate the risks of gradient explosion or vanishing gradients during training. This cautious learning rate enables the model to thoroughly explore the parameter space, reducing the likelihood of missing the global optimum or getting trapped in a local minimum. The model was trained over 180 epochs, allowing sufficient iterations to adapt to the patterns and features within the training data, ultimately enhancing prediction accuracy. The sliding window size was set to 30, a configuration well-suited to the C-MAPSS dataset where engine operating times are recorded at varying timestamps.

Table II. Parameter Setting.

Parameter	Configuration
Window size	30
Learning rate	0.0003
Epoch	180
RUL_{max}	125
Batch size	32
Optimizer	Stochastic gradient descent

In addition to the hyperparameters mentioned above, our study includes setting up a subdomain partitioning network, so the focus is more on finding the optimal subdomain partitioning

density. Therefore, various experiments were conducted utilizing diverse quantities of subdomains, with a thorough analysis of the empirical outcomes. The model performance does not have a positive correlation with the density of subdomain partitioning, as illustrated in Figure 9. The lowest performance was achieved when the number of subdomains was around 10. Therefore, further experiments were conducted, and the optimal partitioning was 11.



(a)Trends in RMSE of different numbers of subdomains (b)Experimental results for the optimal number of subdomains

Fig.9 The impact of the number of subdomains on model performance

Another important factor in the proposed method is the size of the time window. The impact of the time window size on network performance and training time is shown in Figure 10. This result is obtained on the FD001 dataset. The graph distinctly demonstrates that an increased size of the time window results in superior RUL prediction. A larger time window size can include more historical observation data, which can provide richer information to reflect the changes in the device or system's state[48]. By utilizing longer time series data, the prediction model can capture more trends and patterns, thereby making more precise predictions of the remaining useful life and mitigating the influence of noise. If the time window is too large, then it may result in the inclusion of excessive historical data in the model, introducing unnecessary complexity and computational burden. Therefore, when selecting the time window size, achieving a balance between prediction performance and model complexity needs to be considered. When the sliding window increases from 20 to 30, the RMSE parameter optimization is evident, while the training time increases only slightly. However, when the sliding window exceeds 30, the performance stabilizes without significant changes, but the training time increases significantly. Therefore, the size of the sliding window is configured as 30 to maximize the benefits and balance training duration with training effectiveness.

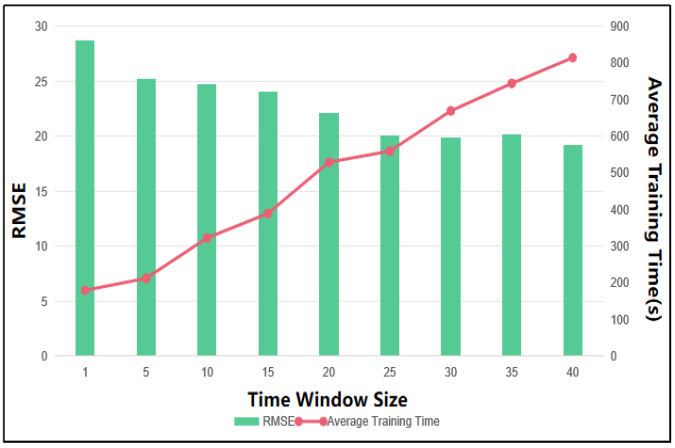


Fig.10 The impact of the time window size on both prognostic performance and computing time during the training process on FD001

4.3. Predictive process

Figure 11 displays the flowchart depicting the proposed prediction method. To begin, we preprocess the C-MAPSS sub-dataset by choosing 14 raw sensor measurement values and normalizing the corresponding data within the range of [−1, 1]. Then, we prepare the dataset for training and testing using a sliding window length to capture time series information within each sample. Notably, the 2D formatted normalized data are directly fed into the model as input, eliminating the need for manual signal processing features such as skewness and kurtosis. As a result, prior expertise in vibration and signal processing is not required to use our proposed method.

Following dataset preparation, we construct the UMDAN network for RUL estimation based on the specific signal processing problem and dataset information, and determine the network's configuration, including the number of hidden layers. UMDAN takes the normalized training data as input and utilizes the labeled RUL values of the training samples as the target output of the network, with backpropagation learning updating the network weights. Each training epoch randomly divides the samples into multiple mini-batches, which are then fed into the training system. After optimization of the network, the model is ready for RUL estimation.

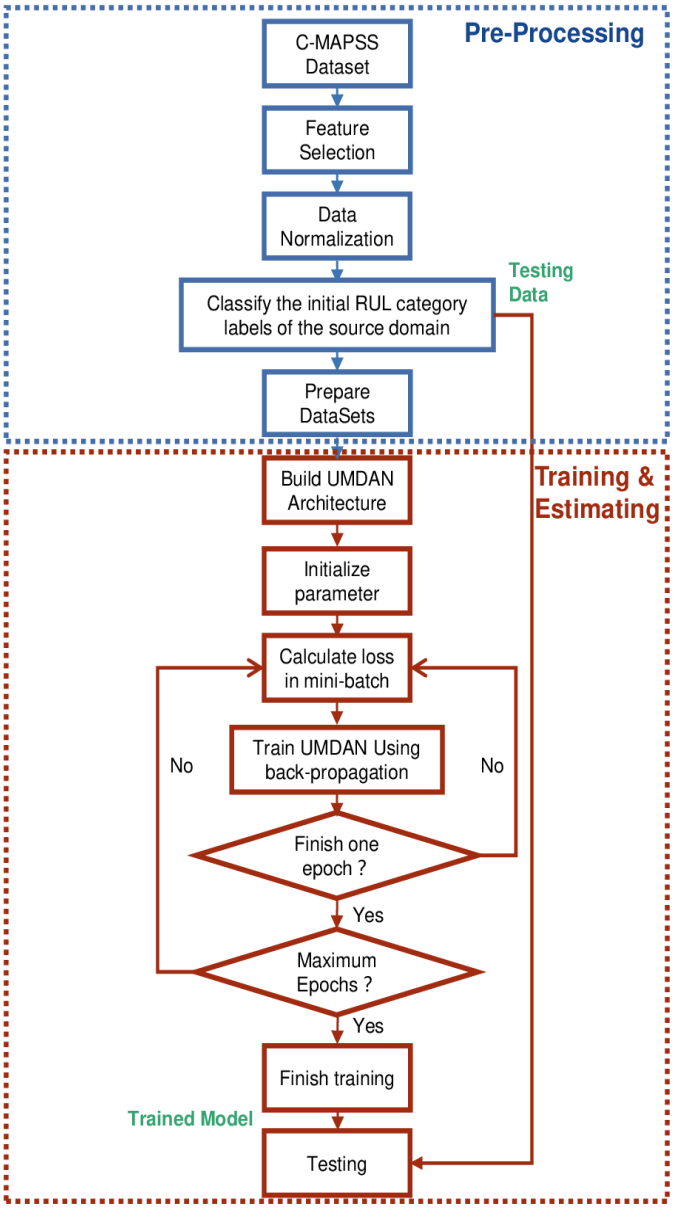


Fig.11. Flow chart of the proposed method for prognostics.

4.4. Method comparison and prognostic performance

To assess the efficiency of the proposed approach, this study designed a total of eight cross-domain transfer tasks, as presented in Table 3. The C-MAPSS dataset consists of four sub-datasets, and each sub-dataset is used as a source task to transfer to the remaining sub-datasets. To holistically assess the model's performance, the prediction performance, and experimental outcomes of the cross-domain transfer tasks, this study utilized two performance evaluation metrics, namely, root mean square error (RMSE) and mean absolute error (MAE), which are expressed as follows:

$$MAE = \frac{1}{M}\sum_{i=1}^{M}|RUL_{pi} - RUL_{ti}| \tag{4.5}$$

$$RMSE = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(RUL_{pi} - RUL_{ti}\right)^2} \tag{4.6}$$

where RUL_{pi} and RUL_{ti} are the RUL benchmark value and the RUL predicted value, respectively. The value of M corresponds to the total number of test samples. A decreased MAE and RMSE signify improved post-prognostics performance of the approach.

Four multisource domain transfer experiments were designed in this paper, and a series of same-method dual-source and single-source domain experiments was set up for comparison to validate the reliability of this method. As shown in the table, in single-source domain transfer, the working conditions in the dataset, failure modes, and engine numbers evidently show that the dataset FD001 is extremely similar to FD003, and the dataset FD002 is extremely similar to FD004. However, in real-world work scenarios, it is not known which data failure modes and other conditions are similar, so multisource domain transfer receives data from multiple domains and is not affected by these conditions. Therefore, in single-source domain transfer, similar domains will not be used for experiments. Detailed information regarding the design of the transfer tasks can be found in Table III. To fully investigate the impact of multi-condition data on model performance, we established three sets of transfer learning tasks based on the C-MAPSS dataset: single-condition data transfer (ST1), dual-condition data transfer (ST2), and triple-condition data transfer (ST3). Both ST1 and ST2 consist of eight subtasks, while ST3 comprises four subtasks. These subtasks utilize the four sub-datasets in C-MAPSS as source and target domains in an alternate fashion.

Table III. Design of transfer tasks based on the C-MAPSS dataset.

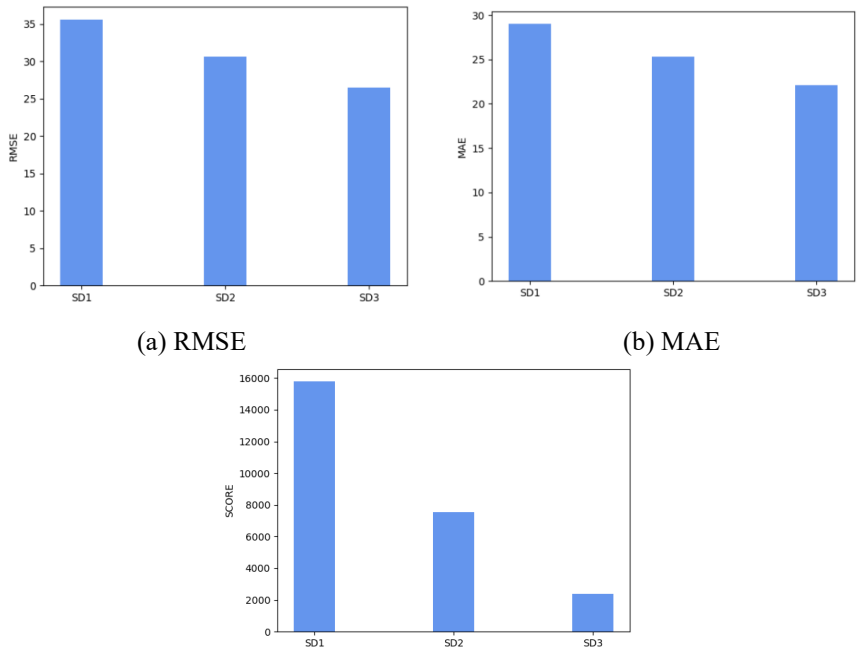
Task	Source	Target
ST1-1	FD002	FD001
ST1-2	FD004	FD001
ST1-3	FD001	FD002
ST1-4	FD003	FD002
ST1-5	FD002	FD003
ST1-6	FD004	FD003
ST1-7	FD001	FD004
ST1-8	FD003	FD004
ST2-1	FD003,FD004	FD001
ST2-2	FD002,FD004	FD001
ST2-3	FD001,FD003	FD002
ST2-4	FD003,FD004	FD002
ST2-5	FD001,FD004	FD003
ST2-6	FD002,FD004	FD003
ST2-7	FD001,FD002	FD004
ST2-8	FD001,FD003	FD004
ST3-1	FD002,FD003,FD004	FD001
ST3-2	FD001,FD003,FD004	FD002
ST3-3	FD001,FD002,FD004	FD003
ST3-4	FD001,FD002,FD003	FD004

Using the proposed method, this study successfully conducted cross-domain prediction tasks outlined in the table. The results of all subtasks for ST1, ST2, and ST3 are displayed in Table IV. From the data in the table, the design of multisource domains has a positive feedback effect on the experimental results. Multisource domain transfer exhibits enhanced average performance when compared with single-source domain transfer.

Table IV. Experimental results of all transfer tasks onC-MAPSS dataset

Task	RMSE	MAE	Score
ST1-1	17.6	14.1	2802
ST1-2	20.4	16.5	2620
ST1-3	39.7	33.7	19425
ST1-4	36.6	30.1	17630
ST1-5	43.6	36.1	23219
ST1-6	37.3	24.3	18951
ST1-7	42.1	37.2	16477
ST1-8	47.1	40.5	25137
ST2-1	18.2	13.5	1806
ST2-2	22.6	19.0	1013
ST2-3	33.5	26.9	10491
ST2-4	19.1	13.7	2207
ST2-5	27.2	22.7	4381
ST2-6	39.2	33.2	26333
ST2-7	41.7	36.0	5187
ST2-8	43.6	37.7	8908
ST3-1	19.9	15.8	1010
ST3-2	25.7	21.2	2669
ST3-3	23.6	18.7	997
ST3-4	36.9	33.0	4921

The average of various experimental result parameters is compared to further explain the impact of multisource domains on the experimental results. Given the lack of control over the influence of the source domain in the single-source domain, the results of all source-to-target domain transfer experiments in the single-source domain are summed and averaged. Similarly, the average of the performance of dual-source and multisource domains is compared to observe the overall performance trends, as depicted in Figure 12.



(a) RMSE (b) MAE

(c) SCORE

Fig. 12. The comparison of RMSE, MAE and SCORE on ST1, ST2 and ST3 tasks.

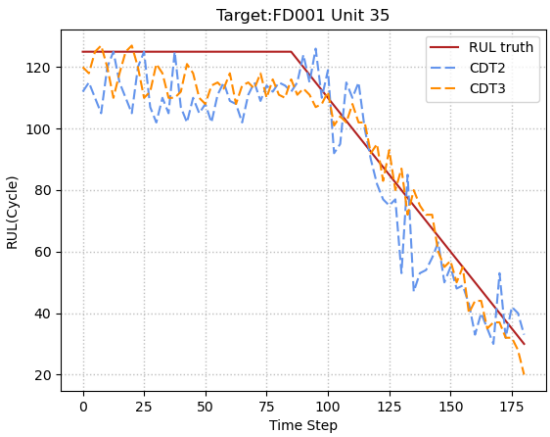
The reduction of the three ST2 metrics by 13.8%, 12.76%, and 52.23%, respectively, as well as the reduction of the three ST3 metrics by 25.35%, 23.79%, and 84.78%, respectively, suggest that our proposed method, which employs data collected under multiple operational conditions for cross-domain RUL prediction tasks, can enhance the precision and adaptability of predictions. The strength of our method is its ability to fully exploit existing datasets and enable target entities to acquire various forms of degradation feature representations. As the data of the target entity is unlabeled, the TL method aims to learn features that are invariant between the source and target domains and transfer the mapping of relationships between the features and labels from the source domain to the target domain. This approach enables the target entity to acquire multiple invariant representations of features, thus resulting in a mapping relationship with the RUL labels that is more accurate and robust compared with using a single feature representation.

Furthermore, the emphasis of the scoring function is on penalizing positive errors, which means that the model's score increases as the model's predicted RUL values surpass the actual RUL values to a greater extent. Practically, this condition results in reduced maintenance time for the system. Evidently, ST2 and ST3 exhibit considerably lower scoring function values in comparison to ST1.

In comparison to the cross-domain prediction task ST1 using single-condition data, multi-condition data leads to a decline in the performance of ST2 and ST3 tasks in RMSE, MAE, and SCORE. ST3 has more abundant source domain data, covering various working conditions and failure modes, which is why its performance is superior to that of ST2. This outcome can be attributed to the fact that the enlarged source dataset boosts the number of training samples, and the diverse degradation features in the target domain can provide a more comprehensive representation of the degradation data.

Consequently, utilizing prediction models that incorporate multi-condition data not only enhances the precision of RUL prediction but also generates RUL forecasts that closely align with real-world scenarios, thereby strengthening system dependability. This capability empowers the system to anticipate failures in advance during operation, consequently mitigating maintenance costs.

In addition, Figure 13 illustrates the RUL predictions of the tested engine units in FD001 before their final recorded cycle. Three instances from a total of 100 experimental engine units in both the dual-source domain and multisource domain are provided, with unit numbers of 22, 35, and 85, respectively. During the initial phases of the three scenarios, the proposed method aims to approximate RUL values that are near a constant RUL_{max}. Subsequently, the estimated values gradually decrease linearly with time until the available test samples are exhausted. Although some conspicuous discrepancies exist between the predicted values and the actual RUL values, the predictive accuracy remains considerably high, especially as the engine units approach failure. This characteristic has significant industrial significance because the late life stage of engine units is pivotal for health management. Effective assessment of the late-life condition of engines can enhance the reliability and safety of engine operation, reduce maintenance costs, and enhance the overall system performance. In addition, on this basis, the fitting effect of the multisource domain is evidently superior to that of the dual-source domain, and the volume of data in the multisource domain has a positive feedback effect on model fitting.



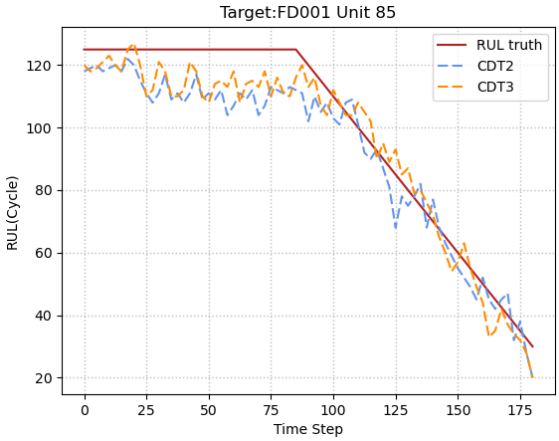


Fig. 13. RUL predictions for unit 22,35and 85 in FD001.

To evaluate the performance improvement of the multisource domain strategy over the single-source domain, this paper compares our proposed method with various transfer methods commonly employed in the realm of RUL prediction. The first method is LSTM-DAAN, which is a multisource domain RUL prediction method rooted in LSTM networks and domain adaptation. Similar to TL-DRR, LSTM-DAAN adapts to the target domain by training on multiple data sources to achieve RUL prediction through domain adaptation. It uses an adversarial loss function that considers domain adaptation and classification errors, which facilitates the preservation of disparate data distributions between the source and target domains. The second method is CNN-DAAN, which, similar to LSTM-DAAN, also uses an adversarial loss function and trains on multiple data sources to adapt to the target domain. CNN-MMD is a multisource domain RUL prediction method based on CNNs and maximum mean discrepancy. It achieves domain adaptation by maximizing the MMD distance between the source and target domains. We compare our proposed multisource domain method with the above methods.

As shown in Table V, our UMDAN approach demonstrates substantial performance gains over these methods with regards to the RMSE metric. In conclusion, the outstanding performance of the UMDAN method regarding the RMSE parameter can be proven to be attributed to its adoption of the multisource domain learning strategy. By training on multiple data sources and utilizing knowledge and feature representations from these sources, UMDAN can capture shared information between different devices more effectively, thus enhancing the precision of RUL estimation.

Table V. Experimental results of Single-source domain methods on C-MAPSS dataset

Task	LSTM-DAAN	CNN-DAAN	TL-DRR	CNN-MMD	**Proposed**
ST1-1	28.1	47.4	34.6	42.4	19.9(ST3-1)
ST1-2	31.5	67.4	35.3	37.2	
ST1-3	48.6	50.3	43.4	49.4	25.7(ST3-2)
ST1-4	44.6	68.2	48.8	62.1	
ST1-5	37.5	40.8	43.2	43.6	23.6(ST3-3)
ST1-6	27.8	45.5	38.7	46.4	
ST1-7	43.8	87.0	45.1	52.5	36.9(ST3-4)
ST1-8	47.9	73.7	52.5	64.0	

The improvement of our method in the multisource domain was validated through experiments using other multisource domain methods on the CMAPSS dataset, comparing the target domains from FD001 to FD004. Two methods were designed, one being the baseline method and the other being a multisource domain transfer method based on DAAN adversarial transfer. To investigate the influence of multisource domain datasets on model performance, this experiment selected the subtask ST3 for comparing experimental results, focusing on the RMSE parameter. The findings presented in Table VI indicate that the performance of the proposed UMDAN method surpasses that of the other two transfer methods by a considerable margin. The UMDAN method performs well whether in the relatively simple working conditions and failure modes in the FD001 and FD003 domains or in the more complex environments of the FD002 and FD004 target domains. Multisource domain datasets strengthen the learning capability of deep features but increase the likelihood of label mapping errors, leading to reduced prediction accuracy. Therefore, the proposed UMDAN method addresses such issues and achieves the lowest RMSE and the highest prediction accuracy. This finding also demonstrates that

the multisource domain adaptive transfer advocated in this paper can improve the precision of RUL prediction by minimizing label mapping errors. In conclusion, the UMDAN method achieves better predictive results, validating its effectiveness and superiority to other single-condition and multi-condition transfer methods.

Table VI. Comparison of performance of different transfer learning methods on various transfer tasks of C-MAPSS dataset.

Target	**Proposed**	Baseline	DAAN
To FD001	19.9	32.1	36.6
To FD002	25.7	48.9	38.7
To FD003	23.6	52.2	45.1
To FD004	36.9	51.6	48.5

On this basis, Figure 14 depicts the test results of the baseline, DAAN, and our proposed methods on the multisource domain target domain FD001. In the figure, the horizontal axis represents the quantity of engine units, while the vertical axis represents the RUL of the engine units. The yellow, gray, and green lines represent the RUL prediction values of UMDAN, DANN, and the baseline, respectively, utilizing input data from the multi-sensor information collected during the final time period of the experimental engine units. The true RUL values of the test engine units are depicted by the red line. Evidently, the UMDAN method, represented by the yellow line, fits the red line better than it fits the other colors.

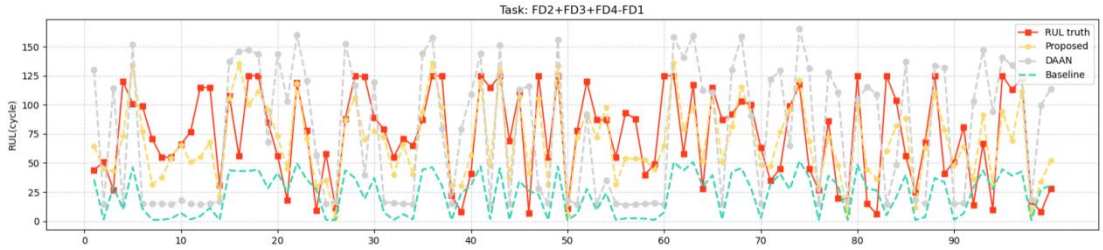


Fig.14. RUL prediction results of three methods on the transfer task ST3-1.

4.5. Ablation experiments

The effectiveness of this approach was verified by designing two ablation experiments that did not fully utilize the proposed method. Method 1 considered only single-subdomain classification adaptive alignment without considering multisource domain transfer, while Method 2 used multisource domain transfer techniques without considering subdomain alignment. Method 3 is to remove the multilinear adjustment part based on method 1. As can be seen from Table VII, both the methods without multisource domain transfer and without subdomain alignment performed worse than the methods that included both, thereby suggesting the efficacy of the proposed experimental methodology. A comparison between the RMSE values of Single-Sub and Single-Multi shows that relying solely on multisource domain transfer without subdomain alignment significantly deteriorated the experimental results. The RMSE value of Single-Sub is significantly better than that of Single-DAAN. This means that the gain to the model from multilinear adjustment is enormous. Although using only subdomain alignment had slightly better results than those of the method only using multisource domain transfer, it still performed much worse than the proposed method did. This finding demonstrates that multisource domain transfer methods are adopted for the generalizability of the model and solve the issue of data scarcity. However, the impact of label alignment errors is significant. Subdomain alignment is a remarkable solution to address multisource domain transfer errors.

Table VII. RMSE of experimental results of ablation experiments on the CMAPSS dataset

Method	Single-Sub	Single-Multi	Single-DAAN	**Proposed**
RMSE	33.4	42.23	47.51	**26.53**

5. Conclusion

In this paper, a transfer model is introduced for cross-condition RUL prediction tasks, employing multisource domain adaptive alignment. By leveraging multiple source domains, the model addresses the limited availability of RUL prediction data and mitigates the potential adverse effects of negative transfer that could arise from relying solely on a single-source domain dataset for transfer learning purposes. By employing a condition-combining DAAN network, which differs from previous adversarial adaptive methods, it addresses the issue of feature and label mapping errors, enhances the integration of condition-aware adversarial domain adaptation, and improves the accuracy of RUL prediction.

It's difficult to accurately obtain source domain data that is exactly similar to the various situations of the target domain in the real production environment even though that it can dramatically improve the model training effect and prediction performance. The multisource domain data makes the experiment more practical in reality. Exhaustive experiments demonstrated significant advantages over popular multisource domain transfer methods in the RUL prediction domain. Still, there are some limitations, the main limitations include dependency on diverse datasets, sensitivity to data noise, and challenges in rapidly changing conditions. Future research should focus on robust noise-handling techniques, adaptive learning mechanisms and comprehensive datasets. Addressing these areas can enhance the model's robustness and applicability in various real-world scenarios.

Reference

1. Kang Z, Catal C, Tekinerdogan B. Remaining Useful Life (RUL) Prediction of Equipment in Production Lines Using Artificial Neural Networks. Sensors 2021; 21:932. https://doi.org/10.3390/s21030932.
2. Zhao B, Ding W, Shan Z, Wang J, Yao C, Zhao Z, Liu J, Xiao S, Ding Y, Tang X, Wang X, Wang Y, Wang X. Collaborative manufacturing technologies of structure shape and surface integrity for complex thin-walled components of aero-engine: Status, challenge and tendency. Chinese Journal of Aeronautics 2023; 36(7): 1-24. https://doi.org/10.1016/j.cja.2023.02.008.
3. Zhao Z, Liang B, Wang X, Lu W. Remaining useful life prediction of aircraft engine based on degradation pattern learning. Reliability Engineering & System Safety 2017; 164: 74-83. https://doi.org/10.1016/j.ress.2017.02.007.
4. Gomes HM, Read J, Bifet A, Barddal JP, Gama J. Machine learning for streaming data: state of the art, challenges, and opportunities. SIGKDD Explorations 2019; 21(2): 6–22. https://doi.org/10.1145/3373464.3373470.
5. Fagogenis G, Flynn D, Lane D. Novel RUL prediction of assets based on the integration of auto-regressive models and an RUSBoost classifier. Proceedings of the International Conference on Prognostics Health Management 2014; 1-6. https://doi.org/10.1109/ICPHM.2014.7036373
6. Tang D, Cao J, Yu J. Remaining useful life prediction for engineering systems under dynamic operational conditions: A semi-Markov decision process-based approach. Chinese Journal of Aeronautics 2019; 32(3): 627-638. https://doi.org/10.1016/j.cja.2018.08.015.
7. Ordóñez C, Lasheras FS, Roca-Pardiñas J, de Cos Juez FJ. A hybrid ARIMA–SVM model for the study of the remaining useful life of aircraft engines. Journal of Computational and Applied Mathematics 2019; 346: 184-191. https://doi.org/10.1016/j.cam.2018.07.008.
8. Sun R. Optimization for Deep Learning: An Overview. Journal of the Operations Research Society of China 2020; 8: 249-294. https://doi.org/10.1007/s40305-020-00309-6
9. Sada SO, Ikpeseni SC. Evaluation of ANN and ANFIS modeling ability in the prediction of AISI 1050 steel machining performance. Heliyon 2021; 7(2): e06136. https://doi.org/10.1016/j.heliyon.2021.e06136.
10. Khelif R, Chebel-Morello B, Malinowski S, Laajili E, Fnaiech F, Zerhouni N. Direct remaining useful life estimation based on support vector regression. IEEE Transactions on Industrial Electronics 2017; 64(3): 2276–2285. https://doi.org/10.1109/TIE.2016.2623260
11. Aggab T, Avila M, Vrignat P, Kratz F. Unifying model-based prognosis with learning-based time-series prediction methods: Application to li-ion battery. IEEE Systems Journal 2021; 15(4): 5245–5254. https://doi.org/10.1109/JSYST.2021.3080125.
12. Li X, Ding Q, Sun JQ. Remaining useful life estimation in prognostics using deep convolution neural networks. Reliability Engineering & System Safety 2018; 172: 1-11. https://doi.org/10.1016/j.ress.2017.11.021.
13. Yang B, Liu R, Zio E. Remaining Useful Life Prediction Based on a Double-Convolutional Neural Network Architecture. IEEE Transactions on Industrial Electronics 2019; 66(12): 9521-9530. https://doi.org/10.1109/TIE.2019.2924605.
14. Lyu Y, Zhang Q, Chen A, Wen Z. Interval Prediction of Remaining Useful Life based on Convolutional Auto-Encode and Lower Upper Bound Estimation. Eksploatacja i Niezawodność – Maintenance and Reliability 2023. https://doi.org/10.17531/ein/165811
15. Zou Y, Donner RV, Marwan N, Donges JF, Kurths J. Complex network approaches to nonlinear time series analysis. Physics Reports 2019; 787: 1-97. https://doi.org/10.1016/j.physrep.2018.10.005.
16. Ma M, Mao Z. Deep-Convolution-Based LSTM Network for Remaining Useful Life Prediction. IEEE Transactions on Industrial Informatics 2021; 17(3): 1658-1667. https://doi.org/10.1109/TII.2020.2991796.
17. Xia T, Song Y, Zheng Y, Pan E, Xi L. An ensemble framework based on convolutional bi-directional LSTM with multiple time windows

for remaining useful life estimation. Computers in Industry 2020; 115: 103182. https://doi.org/10.1016/j.compind.2019.103182.

18. Hsu HY, Srivastava G, Wu HT, Chen MY. Remaining useful life prediction based on state assessment using edge computing on deep learning. Computer Communications 2020; 160: 91-100. https://doi.org/10.1016/j.comcom.2020.05.035.
19. Zhang X, et al. Remaining Useful Life Estimation Using CNN-XGB With Extended Time Window. IEEE Access 2019; 7: 154386-154397. https://doi.org/10.1109/ACCESS.2019.2942991.
20. Zhang A, Wang H, Li S, Cui Y, Liu Z, Yang G, Hu J. Transfer Learning with Deep Recurrent Neural Networks for Remaining Useful Life Estimation. Applied Sciences 2018; 8: 2416. https://doi.org/10.3390/app8122416.
21. Mao W, He J, Zuo MJ. Predicting Remaining Useful Life of Rolling Bearings Based on Deep Feature Representation and Transfer Learning. IEEE Transactions on Instrumentation and Measurement 2020; 69(4): 1594-1608. https://doi.org/10.1109/TIM.2019.2917735.
22. Shen F, Yan R. A New Intermediate-Domain SVM-Based Transfer Model for Rolling Bearing RUL Prediction. IEEE/ASME Transactions on Mechatronics 2022; 27(3): 1357-1369. https://doi.org/10.1109/TMECH.2021.3094986.
23. Zou Y, Li Z, Liu Y, Zhao S, Liu Y, Ding G. A method for predicting the remaining useful life of rolling bearings under different working conditions based on multi-domain adversarial networks. Measurement 2022; 188: 110393. https://doi.org/10.1016/j.measurement.2021.110393.
24. Lyu Y, Wen Z, Chen A. A novel transfer learning approach based on deep degradation feature adaptive alignment for remaining useful life prediction with multi-condition data. Journal of Intelligent Manufacturing 2023. https://doi.org/10.1007/s10845-023-02264-4
25. Yu X, et al. Conditional Adversarial Domain Adaptation With Discrimination Embedding for Locomotive Fault Diagnosis. IEEE Transactions on Instrumentation and Measurement 2021; 70: 1-12. https://doi.org/10.1109/TIM.2020.3031198.
26. Cao Y, Ding Y, Jia M, Tian R. A novel temporal convolutional network with residual self-attention mechanism for remaining useful life prediction of rolling bearings. Reliability Engineering & System Safety 2021; 215: 107813. https://doi.org/10.1016/j.ress.2021.107813.
27. Noh S. Analysis of Gradient Vanishing of RNNs and Performance Comparison. Information 2021; 12: 442. https://doi.org/10.3390/info12110442
28. Yang S, Yu X, Zhou Y. LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example. 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI) 2020: 98-101. https://doi.org/10.1109/IWECAI50956.2020.00027.
29. Bai S, Kolter JZ, Koltun V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. ArXiv, abs/1803.01271.
30. Wang Y, Deng L, Zheng L, Gao RX. Temporal convolutional network with soft thresholding and attention mechanism for machinery prognostics. Journal of Manufacturing Systems 2021; 60: 512-526. https://doi.org/10.1016/j.jmsy.2021.07.008.
31. Ding W, Li J, Mao W, Meng Z, Shen Z. Rolling bearing remaining useful life prediction based on dilated causal convolutional DenseNet and an exponential model. Reliability Engineering & System Safety 2023; 232: 109072. https://doi.org/10.1016/j.ress.2022.109072.
32. Jiang G, Wang J, Wang L, Xie P, Li Y, Li X. An interpretable convolutional neural network with multi-wavelet kernel fusion for intelligent fault diagnosis. Journal of Manufacturing Systems 2023; 70: 18-30. https://doi.org/10.1016/j.jmsy.2023.06.015.
33. Karimian M, Beigy H. Concept drift handling: A domain adaptation perspective. Expert Systems with Applications 2023; 224: 119946. https://doi.org/10.1016/j.eswa.2023.119946.
34. Lu H, Wu J, Ruan Y, Qian F, Meng H, Gao Y, Xu T. A multi-source transfer learning model based on LSTM and domain adaptation for building energy prediction. International Journal of Electrical Power & Energy Systems 2023; 149: 109024. https://doi.org/10.1016/j.ijepes.2023.109024.
35. Chen J, Huang R, Chen Z, Mao W, Li W. Transfer learning algorithms for bearing remaining useful life prediction: A comprehensive review from an industrial application perspective. Mechanical Systems and Signal Processing 2023; 193: 110239. https://doi.org/10.1016/j.ymssp.2023.110239.
36. Ganin Y, Lempitsky V. Unsupervised Domain Adaptation by Backpropagation. Proceedings of the 32nd International Conference on Machine Learning (ICML-15) 2015: 1180-1189.
37. Tzeng E, Hoffman J, Saenko K, Darrell T. Adversarial Discriminative Domain Adaptation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017: 7167-7176. https://doi.org/10.1109/CVPR.2017.316

38. Fu S, Zhang Y, Lin L, Zhao M, Zhong SS. Deep residual LSTM with domain-invariance for remaining useful life prediction across domains. Reliability Engineering & System Safety 2021; 216: 108012. https://doi.org/10.1016/j.ress.2021.108012.
39. Li Y, Liu Y, Zheng D, Huang Y, Tang Y. Discriminable feature enhancement for unsupervised domain adaptation. Image and Vision Computing 2023; 137: 104755. https://doi.org/10.1016/j.imavis.2023.104755.
40. Liu K, Mao W, Zhang W, Wu C, Shi H. An Unsupervised Multi-grade Domain-Adversarial Regression Adaptation Network for Online Remaining Useful Life Prediction of Rolling Bearing Across Machines. 2022 Global Reliability and Prognostics and Health Management (PHM-Yantai) 2022: 1-7. https://doi.org/10.1109/PHM-Yantai55411.2022.9942008.
41. Long M, Cao Z, Wang J, Jordan MI. Conditional adversarial domain adaptation. Proceedings of the Advances in Neural Information Processing Systems 2018: 1640-1650.
42. Asif O, Haider SA, Naqvi SR, Zaki JFW, Kwak KS, Islam SMR. A Deep Learning Model for Remaining Useful Life Prediction of Aircraft Turbofan Engine on C-MAPSS Dataset. IEEE Access 2022; 10: 95425-95440. https://doi.org/10.1109/ACCESS.2022.3203406.
43. Ragab M, Chen Z, Wu M, Kwoh CK, Yan R, Li X. Attention-based sequence to sequence model for machine remaining useful life prediction. Neurocomputing 2021; 466: 58-68. https://doi.org/10.1016/j.neucom.2021.09.022.
44. Zhang C, Lim P, Qin AK, Tan KC. Multi objective deep belief networks ensemble for remaining useful life estimation in prognostics. IEEE Transactions on Neural Networks and Learning Systems 2016; PP(99): 1–13.
45. Yang C, Ma J, Wang X, Li X, Li Z, Luo T. A novel based-performance degradation indicator RUL prediction model and its application in rolling bearing. ISA Transactions 2021. https://doi.org/10.1016/j.isatra.2021.03.045
46. Mo Y, Wu Q, Li X, Huang B. Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit. Journal of Intelligent Manufacturing 2021; 32: 10.1007/s10845-021-01750-x.
47. Wu J, Chen XY, Zhang H, Xiong LD, Lei H, Deng SH. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. Journal of Electronic Science and Technology 2019; 17(1): 26-40. https://doi.org/10.11989/JEST.1674-862X.80904120.
48. Li H, Wang W, Li Z, Dong L, Li Q. A novel approach for predicting tool remaining useful life using limited data. Mechanical Systems and Signal Processing 2020; 143: 106832. https://doi.org/10.1016/j.ymssp.2020.106832. https://doi.org/10.1016/j.ymssp.2020.106832