Strategic Integration of Adaptive Sampling and Ensemble Techniques in Federated Learning for Aircraft Engine Remaining Useful Life Prediction

Ancha Xu[a,b,c], Renbing Wang[b], Xinming Weng[b], Qi Wu[b], Liangliang Zhuang[b,*]

[a] *Economic Forecasting and Policy Simulation Laboratory, Zhejiang Gongshang University, Zhejiang, China*
[b] *School of Statistics and Mathematics, Zhejiang Gongshang University, Zhejiang 310018, China*
[c] *Collaborative Innovation Center of Statistical Data Engineering, Technology & Application Zhejiang Gongshang University, Zhejiang, China*

Abstract

In industrial manufacturing, predicting the remaining useful life of machines is crucial for improving operational efficiency and reducing maintenance costs. However, data privacy concerns and commercial competition make traditional centralized data collection methods insufficient to meet these needs. Federated learning offers a decentralized training approach that protects data privacy, but existing research faces challenges such as inadequate performance of single models, data quality disparities, and improper client selection strategies. To address these issues, this study proposes an adaptive sampling-based ensemble federated learning framework. By integrating the predictions of multiple models, the framework reduces model errors and enhances prediction accuracy and generalization capability. Additionally, we design an adaptive sampling method that dynamically adjusts the client selection strategy based on data quality, focusing on clients with low-quality data to ensure that their contributions are effectively utilized. Experimental results show that the proposed framework significantly outperforms existing benchmark methods on the turbofan engine dataset, with a 12% reduction in RMSE and a 35% decrease in Score. Ablation experiments and sensitivity analysis confirm that the framework maintains reliable predictive performance and efficiency in dealing with issues such as data imbalance, missing data, and scale changes. Supplementary materials for this article are available online.

Keywords: Adaptive sampling; ensemble learning; federated learning; remaining useful life

*Corresponding author
Email address: 22010040022@pop.zjgsu.edu.cn (Liangliang Zhuang)

1. Introduction

Given the increasing complexity and integration of large-scale industrial systems, ensuring the secure and stable operation of systems is crucial in the era of Industry 4.0 [1]. Prognostics and health management (PHM) technology have emerged as a significant solution, attracting extensive attention from researchers and engineers. Predicting the remaining useful life (RUL) of complex systems is a key aspect of PHM, and accurate RUL predictions contribute significantly to proactive maintenance planning, minimizing unforeseen failures, and reducing operational costs [2].

In the realm of RUL prognostics, three primary methodologies are utilized: 1) physics-based methods, 2) data-driven methods, and 3) hybrid methods [3]. Among them, data-driven methods leverage monitoring data to comprehend system degradation trends and extract valuable insights without relying on expert experience [4]. Deep learning (DL) gained significant attention in recent years, particularly in the prognostic domain. DL can effectively employ multi-layered neural networks to autonomously learn intricate representations from data, demonstrating particular prowess in handling complex tasks [5, 6]. Soualhi et al. [7] proposed an RNN-based RUL prediction approach combining direct and recursive methods to address diagnostic uncertainty, demonstrated on a subway door system under varying degradation scenarios and operational conditions. This method is highly adaptive to different degradation scenarios and operational conditions, but its generalization ability is limited, and the prediction accuracy may decline when confronted with new or unseen degradation patterns. Peng et al. [8] proposed a Bayesian DL-based health prognostics method that enhances prediction accuracy through uncertainty quantification. Its effectiveness was validated using datasets from rolling bearings and turbofan engines. This method's advantage lies in its ability to quantify uncertainty, thereby enhancing the reliability of predictions. However, its main limitation is the high demand for quality data, which restricts its applicability in data-scarce scenarios. Xu et al. [9] developed a hybrid deep learning model integrating handcrafted features, domain knowledge, and latent features extracted via DL networks to improve early RUL prediction accuracy. Although this method improves accuracy, its limited generalization ability and high computational complexity remain challenges, particularly when dealing with new degradation patterns or incomplete data, where its performance may degrade. Table 1 provides a comparative summary of existing RUL prediction methods. Further advancements in DL-based prognostics can be found in studies like [10, 11, 12, 13, 14, 15, 16].

Table 1: Comparative Analysis of RUL Prediction Methods under Centralized Learning Pattern.

Method	Type	Model	Dataset	Strengths	Weaknesses
[17]	PB	Finite element	Bearing	Strong interpretability	High complexity
[18]	DD-SM	Stochastic process	Battery	Uncertainty modeling	Strong assumptions
[7]	DD-DL	RNN	Servomotor	Diversity adaptation	Limited generalization
[8]	DD-DL	BDL	Bearing, engine	Uncertainty modeling	High-quality data
[9]	Hybrid	CNN with knowledge	Battery	Improved accuracy	Limited generalization

Note: "PB" means physics-based"; "DD" means "data-driven"; "SM" means "statistical model"; "DL" means "deep learning".

Data-driven prognostic approaches have shown promising results, but they heavily depend on high-quality training data, which is often scarce in industrial settings. The limited availability of run-to-failure data and the high labor and costs associated with independent model development further exacerbate the issue. Additionally, while companies use similar machinery, data aggregation is hindered by concerns over trade secrets and conflicts of interest, creating what is known as the isolated data island problem [19]. This dispersal of data makes centralized model training impractical. Decentralized approaches like federated learning (FL) [20] offer a solution by enabling collaborative model training across multiple local clients while preserving data privacy. With increasing concerns over security and privacy, FL has gained traction in fields such as financial security, manufacturing, and healthcare [20, 21].

In recent years, an increasing number of scholars have been applying FL methods to RUL prediction [22, 23]. Table 2 summarizes key FL approaches in this domain, highlighting their strengths and weaknesses. For example, Zhang et al. [24] proposed a novel FL framework for predicting wind turbine blade icing, integrating human expertise in feature selection, and a method for addressing class imbalance. The framework demonstrates strong robustness and convergence under significant data heterogeneity, but its reliance on a single deep learning model limits its adaptability. Guo et al. [25] proposed an FL approach for estimating the lifespan of milling cutters. In this method, the cloud server assigns weights to each client based on the convolutional autoencoder (CAE) reconstruction error, while the prediction model is centrally trained by the server. This approach reduces computational burden on edge devices, but its dependence on a single CAE architecture limits its

applicability across diverse operational scenarios. Du et al. [26] developed a lightweight FL model utilizing a transformer encoder, achieving minimal prediction errors. By omitting the decoder module, the model reduces computational and memory overhead; however, its aggregation performance may degrade when faced with highly data distribution imbalances. Kamei and Taghipour [16] applied the FL framework to predict the RUL of turbofan engine data and conducted a comprehensive comparison of its predictive performance with centralized learning methods. The method utilizes LSTM and Transformer, demonstrating strong predictive performance. However, the simplistic sampling and aggregation strategies may limit the model's generalization ability, particularly when dealing with imbalanced or missing data. Zhang et al. [27] developed an FL-based multi-hop graph pool adversarial network for RUL prediction, applied to turbine engines and bearing datasets. The method demonstrates good robustness when handling data from different sources. However, its multi-module design increases computational and storage demands, limiting its application on resource-constrained devices. While the aforementioned studies on FL have made valuable contributions, they also exhibit several common shortcomings:

(i) **Limitations of single models**: These models rely on a single DL approach for RUL prediction within the FL framework. However, their practicality is limited under complex industrial conditions [28]. Key issues include: i) Difficulty in adapting to diverse operating conditions in industrial systems, such as temperature fluctuations, humidity changes, and load variations [29]. This lack of adaptability to changing conditions may lead to a decline in model performance. ii) Sensitivity to noise and missing data, which can cause the model to become unstable when handling real-world industrial environmental data [30, 31].

(ii) **Limitations of data quality and client selection**: Variations in data sources or collection methods result in significant disparities in data quality and availability among clients. Most works in Table 1 use a strategy that involves all clients in each training round, ensuring data diversity but also leading to high communication overhead and computational cost. More importantly, these methods often neglect dynamic adjustments based on client data quality, thereby failing to optimize the training process effectively. Thus, more efficient client selection strategies are needed to reduce training time while maintaining accuracy.

Table 2: Comparative analysis of FL methods in RUL prediction.

Method	Data handling	Model	FL strategy	Strengths	Weaknesses	Dataset
[24]	Oversampling to alleviate class imbalance	LSTM, CNN	Randomly select 50% clients; weighted aggregation by data size, timestamp	Improved robustness and convergence under data heterogeneity	Single model	Wind turbine
[25]	Multi-scale learning, self-attention	Convolutional autoencoder	All clients; weighted aggregation based on validation	Lightweight client-side computation; reduces edge load	Single CAE model	Milling cutter, XJTU bearing
[26]	Sliding window, multi-head self-attention	Transformer encoder, FFN	All clients; FedAvg with weighted aggregation based on data volume	Reduced overhead with simplified decoder, suitable for resource-constrained devices	Data distribution imbalances may impact aggregation effectiveness	C-MAPSS engine
[16]	None	LSTM, Transformer	All clients; FedAvg, FedProx	Transformer framework for RUL prediction	Simple aggregation; limited generalization	C-MAPSS engine
[27]	Multi-hop graph pooling	GNN, adversarial transfer learning	All clients; dynamic weighted aggregation by loss	Reduces domain shift via adversarial learning	High computation and storage demand	C-MAPSS engine, XJTU bearing

Table 3: Comparison of recent FL-based methods.

Method	Client selection	Training model	Applicaiton
Zhang et al. [24]	Random	Single	RUL prediction
Guo et al. [25], Du et al. [26], Kamei and Taghipour [16], Zhang et al. [27]	All	Single	RUL prediction
Wang et al. [32]	Random	Ensemble	Decision optimization
Shi et al. [33]	All	Ensemble	Classification tasks
AS-EFL	Adaptive	Ensemble	RUL prediction

To address the aforementioned limitations, we propose a novel FL approach for RUL prediction. The key contributions and innovations are as follows: First, we introduce an **a**daptive **s**ampling-based **e**nsemble **FL** approach, named **AS-EFL**, designed to handle data quality variance among clients and the instability of single-model predictions under complex conditions. Table 3 compares recent FL-based methods. To the best of our knowledge, this is the first study to integrate adaptive sampling (AS) and ensemble algorithms into FL for RUL prediction. Specifically, we employ ensemble learning to stabilize model performance in the presence of noise and missing data, effectively combining global information to enhance accuracy and generalization. This ensemble approach can be used with any FL strategy without altering the aggregation of global models. This concept has been successfully applied to decision optimization [32] and classification tasks [33] within the FL framework. Additionally, we propose the AS method to address the second limitation of FL. The AS module dynamically focuses on clients with poorer data quality, ensuring their contributions are effectively utilized. Furthermore, we apply the proposed model to a turbofan engine dataset and conduct a case study. Comparative analysis with several benchmark strategies confirms the superior performance of our method in predicting RUL.

The remainder of the paper is structured as follows: Section 2 introduces the AS-EFL framework for RUL prediction, providing detailed insights along with the algorithmic workflow of the proposed model. Section 3 presents a comprehensive case study to validate the proposed methodology. Finally, Section 4 concludes the paper with a summary of our findings.

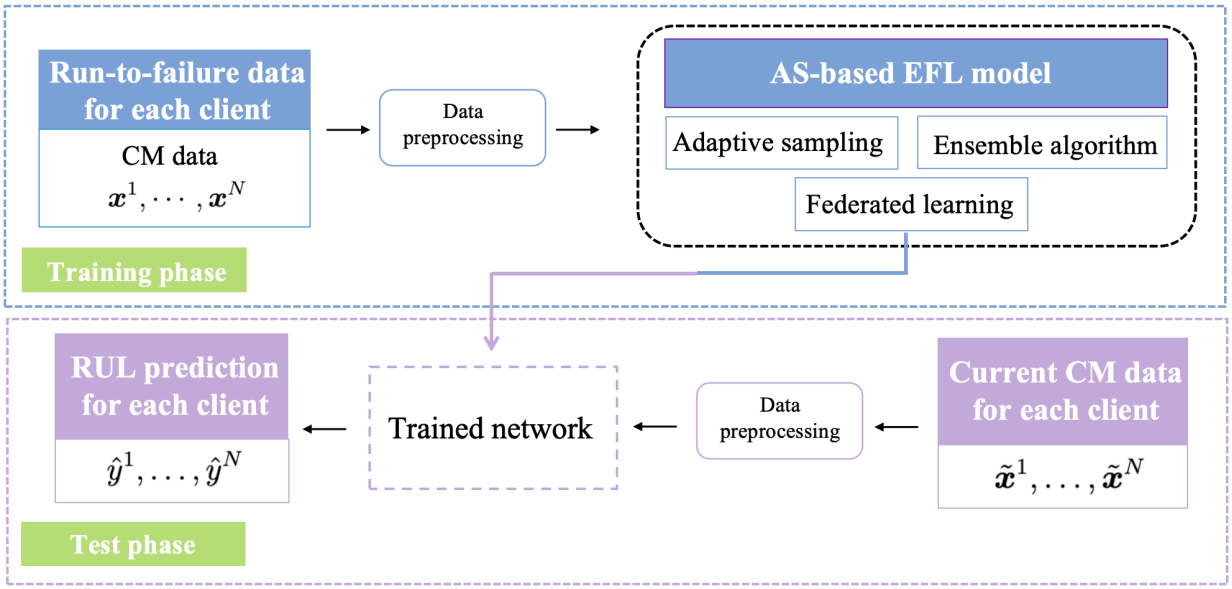


Figure 1: Proposed AS-EFL framework for RUL prediction.

2. AS-EFL framework for RUL prediction

In this section, we first introduce the AS-EFL framework for RUL prediction. Suppose we have N clients, the private data set of the k-th client is denoted as $\mathcal{D}_k = \{(\boldsymbol{x}_i^k, y_i^k)\}_{i=1}^{n_k}$, where n_k is the number of test systems in the k-th client. Each client's dataset contains condition monitoring (CM) data $\boldsymbol{x}^k = \{\boldsymbol{x}_1^k, \ldots, \boldsymbol{x}_{n_k}^k\}$, and corresponding RUL, $y^k = \{y_1^k, \ldots, y_{n_k}^k\}$. To protect client privacy, we assume that local private datasets are not accessible between different clients. Our objective is to construct a global predictive model $y = F(\boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents the parameters within the FL framework without explicitly centralizing the dataset $\boldsymbol{\mathcal{D}} = \{\mathcal{D}_k, k = 1, 2, \cdots, N\}$.

The proposed RUL prognostic framework consists of training and test phases (Figure 1). During the training stage, efforts are devoted to developing a global model—utilizing run-to-failure data from each client—to map historical CM data to RUL. To achieve this objective, following the preprocessing of the collected CM data on each client, an AS-based EFL model is trained on the server for RUL prediction. Firstly, we introduce the traditional FL algorithm for decentralized training (see Section 2.1), and emphasize the problems of FL in RUL prediction. To address the instability of single-model prediction, we utilize ensemble learning to aggregate the outputs of multiple DL networks, resulting in the final trained model (see Section 2.2). To mitigate the poor aggregation performance and long

training times of traditional FL, we select client data adaptively based on its quality (see Section 2.3). At the test phase, current CM data $\{\tilde{\boldsymbol{x}}^1, \ldots, \tilde{\boldsymbol{x}}^N\}$ is collected from each client. Following pre-processing, the selected data are inputted into the trained network for RUL prediction on each client.

2.1. Federated learning

Figure 2 depicts the comprehensive architecture of an FL scheme, comprising local clients and a cloud server. The key steps of FL are as follows: initially, local clients train their models using training CM data obtained from sensors. Subsequently, these local models are updated to the server and aggregated to construct a global model. Then, the aggregated global model, along with parameters, is distributed to all clients for their own tasks. Repeat the above steps until predefined termination criteria are met (e.g., reaching maximum iteration count or model accuracy exceeding a threshold). The central server aggregates updates and ultimately determines the global model.

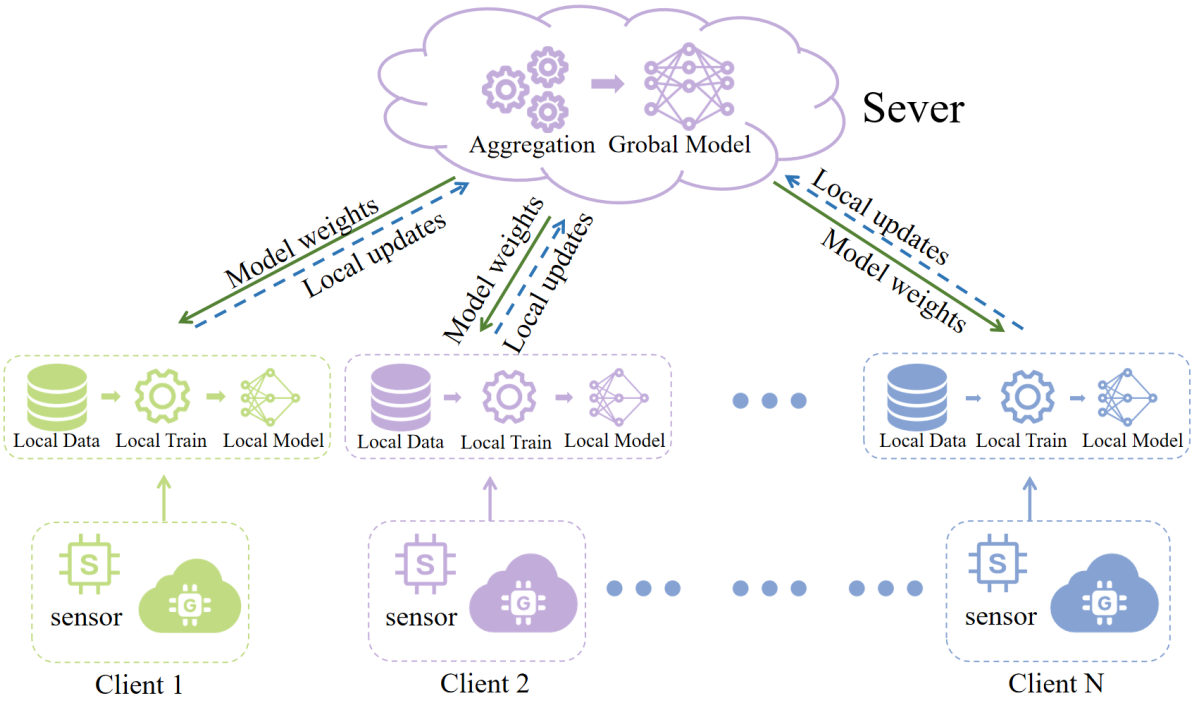


Figure 2: Illustration of the FL scheme: decentralized model training and aggregation process.

Specifically, the learning process of FL is driven by minimizing the global loss function,

which is calculated on each client using a weighted aggregation method:

$$\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) = \sum_{k=1}^{N} p_k F_k(\boldsymbol{\theta}) = \mathbb{E}\left[F_k(\boldsymbol{\theta})\right], \tag{1}$$

where p_k is the probability/weight of the k-th client proportional to the local data size n_k, i.e., $p_k = n_k/n$, where n is the total data size across all clients $n = \sum_{k=1}^{N} = n_k$. It verifies $p_k \geq 0$ and $\sum_{k=1}^{N} p_k = 1$. Clearly, if all clients have the same dataset size n_i, objective (1) reduces to $\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) = 1/N \sum_{i=1}^{N} F_k(\boldsymbol{\theta})$. Here, $F_k(\boldsymbol{\theta})$ represents the local loss function, defined as

$$F_k(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}^k}\left[f_k\left(\boldsymbol{\theta}; \boldsymbol{x}^k\right)\right] = \sum_{i=1}^{n_k} f_k\left(\boldsymbol{\theta}; \boldsymbol{x}_i^k\right), \tag{2}$$

where $f_k(\boldsymbol{\theta}; \boldsymbol{x})$ is the loss incurred by predicting on sample $\boldsymbol{x}$ using model $\boldsymbol{\theta}$ at client k. FL attempts to mitigate the average risk of clients by facilitating independent risk function calculation for each client, unlike traditional centralized approaches, thus eliminating the need for data aggregation [34].

Commonly used FL algorithms include federated averaging (FedAvg) and federated proximal term (FedProx). The advantage of these algorithms lies in their ability to effectively coordinate model updates from various devices while preserving data privacy, thereby improving the performance and reliability of the global model. These algorithms have been widely applied in various fields such as healthcare [35], energy forecasting [19], and transportation management [36]. Next, we will detail the training process of FL based on these two algorithms. It constitutes the following steps:

Step 1: client selection and model broadcasting: the server randomly selects S clients from N devices according to selection probabilities p_k, and the corresponding subset is $\mathcal{S}_t$. Then, the server transmits the current global model $\boldsymbol{\theta}^t$ during communication round t to local clients. In the case of initial communication, the model assigned by the server is denoted as $\boldsymbol{\theta}^0$, which can be determined by the engineer.

Step 2: local training: for FedAvg algorithm, $\boldsymbol{\theta}_k^{t+1}$ is calculated through multiple stochastic gradient descent (SGD) iterations with learning rate η based on minimizing the loss function:

$$\boldsymbol{\theta}_k^{t+1} \approx \arg\min_{\boldsymbol{\theta}} F_k(\boldsymbol{\theta}). \tag{3}$$

However, when strong statistical heterogeneity is present, i.e. F_k is different than f, the local model may converge to F_k and diverge from the global model f. To overcome this

Algorithm 1: The FL framework based on FedAvg or FedProx.

Input: $\mathcal{D}, \boldsymbol{\theta}^0, T, \eta, N, S$, and $p_k, k = 1, \ldots, N$.

Output: $\boldsymbol{\theta}^*$.

```
for t = 1 to T do
    The server randomly selects a subset S_t, and sends θ^t to clients;
    for k ∈ S_t do
        Compute θ_k^{t+1} based on FedAvg (3) or FedProx (4);
        Update θ_k^{t+1} to the server.
    end
    The server obtains θ^{t+1} using (5);
end
Denote θ* = θ^T as the optimal model, and sends θ* to all clients.
```

problem, FedProx adds a regularization term based on (3) to keep the local model close to the current model. The $\boldsymbol{\theta}_k^{t+1}$ is obtained by

$$\boldsymbol{\theta}_k^{t+1} \approx \arg\min_{\boldsymbol{\theta}} \left\{ F_k(\boldsymbol{\theta}) + \frac{\mu}{2} \left\| \boldsymbol{\theta} - \boldsymbol{\theta}^t \right\|^2 \right\}, \tag{4}$$

where μ is the proximal hyperparameter and needs to be determined.

Step 3: model updates and server aggregation: Then each selected user sends its updated model to the server for aggregation:

$$\boldsymbol{\theta}^{t+1} = \frac{1}{S} \sum_{k \in \mathcal{S}_t} \boldsymbol{\theta}_k^{t+1}. \tag{5}$$

Continue iterating through the aforementioned steps until the number of epochs reaches T. At this point, the server broadcasts the optimal/final model $\boldsymbol{\theta}^*$ to all clients. Subsequently, each local client conducts tasks like predicting RUL and formulating maintenance strategies based on the latest CM data, utilizing the global model. The FL framework based on FedAvg or FedProx is outlined in Algorithm 1.

2.2. Ensemble learning

Traditional FL methods primarily rely on a single deep learning approach for local model training. However, their practicality may be limited under complex industrial conditions. Ensemble learning, by fusing predictions from multiple models, effectively incorporates global information and improves prediction accuracy [37]. The underlying idea is that

the errors of individual models can be compensated by others, thereby enhancing overall predictive performance [31]. This approach has been successfully applied to RUL prediction in studies such as [31] and [28]. Key considerations in this approach include the selection of member models and the choice of fusion strategy.

2.2.1. Member predictor selection

We use DL methods to train CM data and build RUL prediction models. The commonly used methods in RUL prediction are based on foundational network architectures such as:

- Convolutional neural network (CNN): It excels at capturing local and global features within structured data. To adapt CNNs for time-series data, we utilize multi-dimensional convolutional neural networks (DCNNs), which can effectively capture local patterns and temporal dependencies [38, 39].
- Recurrent neural network (RNN): RNNs are designed to process sequence data. Their improved variants, Long short-term memory (LSTM) and gated recurrent unit (GRU), address issues like vanishing gradients and long-term dependencies. LSTM has stronger memory capabilities, while GRU provides faster training with fewer parameters [40, 41].

These models are selected as member predictors for their complementary characteristics in handling time-series data. While the ensemble approach increases computation and communication overhead compared to single-model FL (as each client trains and sends updates for multiple models), it can address challenges like data imbalance, missing data, and noise, enhancing the global model's generalization and stability. Note that engineers can modify these models by adding layers or adjusting architectures to specific datasets or domain knowledge, further improving adaptability.

2.2.2. Fusion strategy selection

After selecting the member predictors, ensemble learning methods fusion their predictions to generate the final output. For regression tasks like RUL prediction, weighted averaging is commonly used. Instead of relying on computationally intensive statistical methods like linear regression or ridge regression [31], we propose a dynamic weight adjustment strategy based on a scoring function, which allows updates during FL iterations. In this context, the scoring function evaluates each predictor's performance on the local dataset

during training. Suppose we use M predictors for RUL prediction. The scoring function for the m-th predictor in the k-th client is defined as:

$$\mathrm{SC}_{m,k} = \sum_{i=1}^{n_k} s_{m,i}, \text{ where } s_{m,i} = \begin{cases} e^{-\frac{d_{m,i}}{13}} - 1, & \text{if } d_{m,i} < 0, \\ e^{\frac{d_{m,i}}{10}} - 1, & \text{if } d_{m,i} \geq 0. \end{cases} \tag{6}$$

Here $d_{m,i}$ represents the difference between the estimated and actual RUL values for the i-th instance. A smaller value indicates a better fit of the model. During each local training, we first compute the $\mathrm{SC}_{m,k}$ and then perform the following steps:

(i) Dynamic weight computation: the weights $\omega_{m,k}$ for the m-th predictor on the k-th client are calculated based on the inverse of the scoring function, as follows:

$$\omega_{m,k} = \frac{1/\mathrm{SC}_{m,k}}{\sum_{m=1}^{M} 1/\mathrm{SC}_{m,k}}. \tag{7}$$

These weights are normalized scores, ensuring that the contribution of each predictor reflects its relative performance. Predictors with smaller $\mathrm{SC}_{m,k}$ (indicating better performance) will have higher weights in prediction fusion.

(ii) Weight update and fusion: In each communication round t, the model parameters $\boldsymbol{\theta}_{m,k}^t$ for all M predictors are uploaded from the k-th client to the server. The server aggregates these parameters for each predictor m as:

$$\boldsymbol{\theta}_m^t = \frac{1}{S} \sum_{k \in \mathcal{S}_t} \boldsymbol{\theta}_{m,k}^t, \quad m = 1, \ldots, M. \tag{8}$$

Notably, the local weights $\omega_{m,k}$ remain on the client side and do not need to be transmitted.

The above process is repeated until the stopping criterion is met. After training, the server broadcasts the aggregated model parameters $\boldsymbol{\theta}_m^*$ back to the clients. Upon receiving the final global model parameters, each client fuses the predictions from its ensemble predictors using the locally stored optimal weights $\omega_{m,k}^*$. For a new dataset $\tilde{\boldsymbol{x}}^k$, the final RUL prediction is computed as:

$$\hat{y}_{\mathrm{es}}^k = \sum_{m=1}^{M} \omega_{m,k}^* \hat{y}_m^k, \tag{9}$$

where $\hat{y}_m^k$ is the prediction of the m-th predictor for the k-th client.

Figure 3 provides a visual overview of the proposed ensemble learning framework within the FL context. The diagram illustrates how individual predictors operate independently at the client level, their parameter aggregation at the server, and the final ensemble-based prediction process using dynamically updated weights.

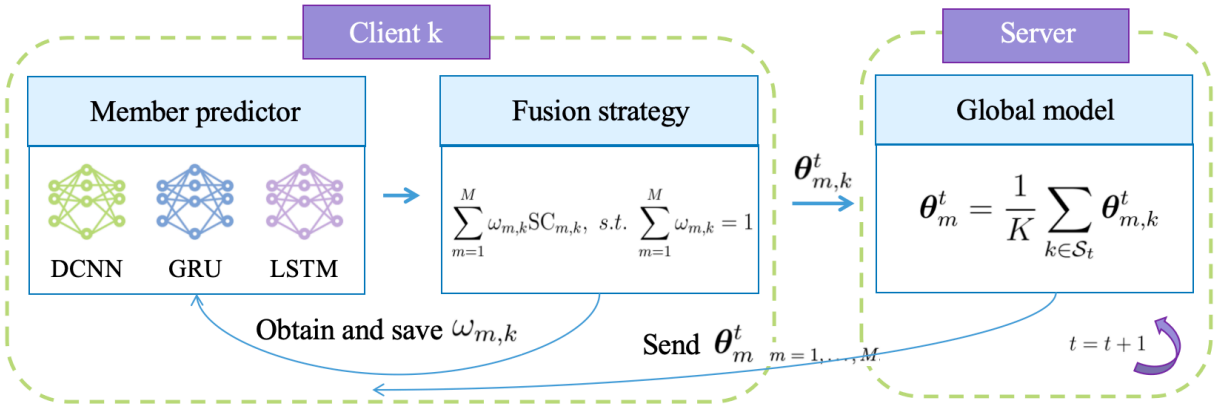


Figure 3: Ensemble framework based on FL for RUL prediction.

2.3. Adaptive sampling

In algorithm 1, the server selects a subset $\mathcal{S}_t$ of N clients in each communication round according to a fixed probability p_k. However, the fixed selection probability may not adequately account for clients with poor data quality. To improve the global model's performance, more attention should be given to these clients to ensure they contribute meaningfully to the global model update. Based on this, we propose an adaptive sampling method that dynamically updates the sampling probabilities $p_{m,k}^{t}$ for each client after each communication round. Specifically, for the k-th client and m-th predictor, the sampling probability $p_{m,k}^{t+1}$ is calculated as:

$$p_{m,k}^{t+1} = \frac{\beta_{m,k}^{t}}{\sum_{k=1}^{N} \beta_{m,k}^{t}}, \quad \text{where} \quad \beta_{m,k}^{t} = \exp\left(\Phi_{m,k}^{t}\right). \tag{10}$$

Here $\beta_{m,k}^{t}$ is a client-specific weight, which is computed based on the performance metric $\Phi_{m,k}^{t}$. For example, we use either root mean square error (RMSE) or relative bias (RB) as evaluation metrics, and their formulas are as follows:

$$\mathrm{RMSE}_{m,k}^{t} = \sqrt{\frac{\sum_{i=1}^{n_k} \left(d_{m,i}\right)^2}{n_k}}, \quad \text{or} \quad \mathrm{RB}_{m,k}^{t} = \frac{1}{n_k} \sum_{i=1}^{n_k} d_{m,i}. \tag{11}$$

Smaller values of RMSE or RB indicate lower error and bias in the client's predictions. With these performance metrics in place, the server can adaptively adjust the sampling probabilities based on each client's performance.

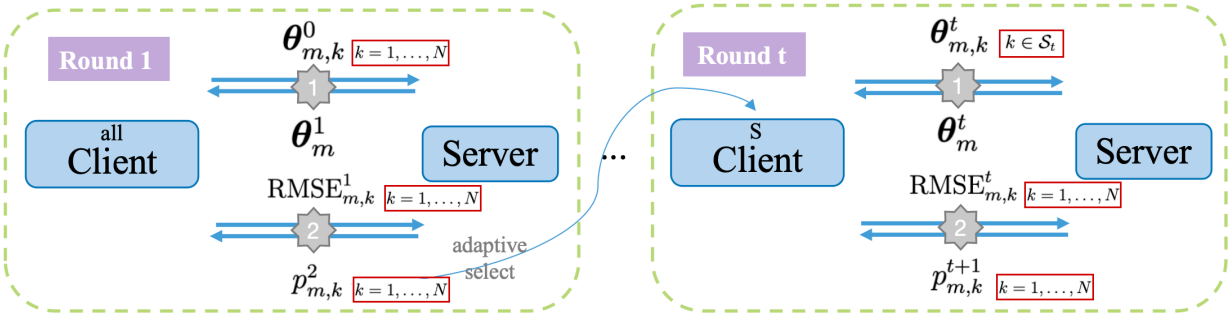


Figure 4: AS framework update process.

Next, we describe the adaptive update process of the AS strategy, as shown in Figure 4. In the first communication round, all clients participate in training with equal initial sampling probabilities ($p^0_{m,k} = 1$). Each client trains its model based on local data, computes model parameters $\boldsymbol{\theta}^0_{m,k}$, and sends them to the server. The server aggregates the received parameters and updates the global model parameters $\boldsymbol{\theta}^1_m$, which are then redistributed to all clients. Each client uses the updated global model to make predictions, computes the corresponding performance metric (either $\text{RMSE}^1_{m,k}$ or $\text{RB}^1_{m,k}$), and sends the results back to the server. The server then updates the sampling probabilities $p^2_{m,k}$ for the next round. In subsequent rounds, S clients are selected for local model training based on the updated probabilities. The other processes are similar to the first round, and we will not elaborate further. This process is repeated until the predetermined number of communication rounds is reached.

2.4. Computational complexity analysis

The AS-EFL framework for RUL prediction is implemented as outlined in Algorithm 2. In each round t, the server selects S clients based on their sampling probability p^t_k. Each selected client then updates M models locally, with the update complexity of $O(u)$ per model. Therefore, the total complexity for each client in one round is $O(SMu)$. After local updates, the clients upload their M model parameters (dimension d) for aggregation, which has a complexity of $O(SMd)$. The server then distributes the aggregated parameters to all N clients, with a complexity of $O(Nd)$. Each client computes and uploads the evaluation

matrix based on the aggregated parameters $\boldsymbol{\theta}_{\boldsymbol{m}}^{\boldsymbol{t}}$, with a complexity of $O(NMv)$. The server updates the sampling probabilities $p_{m,k}^{t+1}$ and distributes them to the clients, which has a complexity of $O(NM)$. Therefore, the total time complexity for one round is $O(SMu + SMd + Nd + NMv + NM)$.

Regarding space complexity, each client needs to store M models, each with parameters of dimension d, resulting in a storage complexity of $O(Md)$ per client. For all N clients, the total storage complexity is $O(NMd)$. Additionally, each client stores its own data, with the total storage for all clients being $O\left(\sum_{k=1}^{N} \varsigma_k\right)$, where ς_k denotes the data size of client k. On the server, the storage for aggregated parameters is $O(Md)$, and for sampling probabilities, it is $O(NM)$. Thus, the total space complexity of the system is $O(NMd+\sum_{k=1}^{N} \varsigma_k+Md+NM)$. Section 3.6.3 will present real-data experimental results and evaluate the computational efficiency at different data scales.

3. Experimental study

In this section, the proposed AS-EFL method is applied to a turbofan engine dataset. Section 3.1 provides an overview of the dataset and the preprocessing techniques applied. Section 3.2 introduces several benchmark models and outlines the experimental setup. Section 3.3 presents the implementation process of the proposed model. Section 3.4 comprehensively demonstrates the predictive accuracy of the proposed model compared to the other benchmark models. Sections 3.5 and 3.6 provide the results of the ablation study and sensitivity analysis, respectively.

3.1. Dataset description and preprocessing

We validate the effectiveness of the AS-EFL framework for RUL prediction using the well-known C-MAPSS dataset [42]. This dataset is generated within a MATLAB-Simulink virtual environment from the NASA Ames Prognostics Center of Excellence. For our study, we use the "FD001" sub-dataset, which contains CM data from 100 engines operated under the same conditions and a single fault mode. The CM data from 100 engines running until failure are designated as the training dataset, while the CM data from the remaining 100 operational engines are used as the test dataset.

These CM data comprise 21 sensor-acquired measurements, with some sensor data exhibiting minimal variation and offering negligible contributions to RUL prediction. To enhance computational efficiency and reduce training time, these features are eliminated.

Algorithm 2: The AS-EFL framework for RUL prediction.

Input: $\mathcal{D}, \boldsymbol{\theta}_m^0, T, \eta, N, M, S, \tilde{\boldsymbol{x}}^k$, and $p_{m,k}^0, k = 1, \ldots, N$.

Output: $\boldsymbol{\theta}_m^*, \omega_{m,k}^*, \hat{y}_{\text{es}}^k, m = 1, \ldots, M$, and $k = 1, \ldots, N$.

for $t = 1$ **to** T **do**
  The server selects $\mathcal{S}_t$ clients based on p_k^t, and sends $\boldsymbol{\theta}_m^t$ to clients.
  for $k \in \mathcal{S}_t$ **do**
    for $m = 1$ **to** M **do**
      Calculate $\boldsymbol{\theta}_{m,k}^{t+1}$ based on (3) or (4), and update it to server;
    end
  end
  The server obtain $\boldsymbol{\theta}_m^{t+1}$ using (8) and sends it to clients.
  for $k = 1$ **to** N **do**
    for $m = 1$ **to** M **do**
      Based on $\boldsymbol{\theta}_m^{t+1}$, compute $\text{RMSE}_{m,k}^t$ or $\text{RB}_{m,k}^t$ using (11), and send to server.
    end
  end
  Update and save sampling probability $p_{m,k}^{t+1}$ using (10);
end
Denote $\boldsymbol{\theta}_m^* = \boldsymbol{\theta}_m^T$, and send well-trained model $\boldsymbol{\theta}_m^*$ to all clients;
Denote $\omega_{m,k}^* = \omega_{m,k}^T, m = 1, \ldots, M$, and $k = 1, \ldots, N$;
Based on the well-trained model, calculate $\hat{y}_{\text{es}}^k$ based on new CM data $\tilde{\boldsymbol{x}}^k$ with (9).

Following the approach in Li et al. [38], 14 sensor measurements, indexed as 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, and 21, are used as the original input features. The CM data from each sensor undergo normalization to fall within the range of $[-1, 1]$, achieved through the min-max normalization method. To augment the training dataset, a sliding time window approach is adopted [43]. By comparing the model's predictive performance with different window sizes (detailed results are provided in Supplementary Section S1), we find that a window size of 30 produced the best results. Therefore, 30 consecutive points are selected as the input sample size for the proposed model. For label rectification, a linear RUL function with a maximum value of 125 is applied to each training sample [44, 38].

To simulate the FL scenario, we randomly partition 100 engines into different clients,

assuming no communication between these clients. We consider three scenarios with different numbers of clients, denoted as $N = 3$, 5, or 8, aiming to distribute an equal number of test engines to each client whenever possible [26, 16]. Additionally, in Section 3.6.1, we also investigate scenarios where there are substantial differences in sample sizes among clients and compare the predictive performance of different models. We consider two commonly used FL algorithms, namely FedAvg and FedProx, with hyperparameter settings detailed in Table 4. The choice of hyperparameters is based on balancing model performance and computational efficiency. Batch size is set to 64 or 128 depending on the client capacity, with 64 providing a good trade-off between stability and efficiency, while 128 is used for larger client configurations to speed up convergence. Epochs represents the number of iterations conducted by each local client during model training and is set to 50 or 80 to ensure sufficient training without overfitting. T is the number of communication rounds, chosen as 5 or 8 to balance communication cost and convergence speed. For μ in FedProx, a value of 0.01 is selected to mitigate the impact of non-IID data without over-regularizing the model.

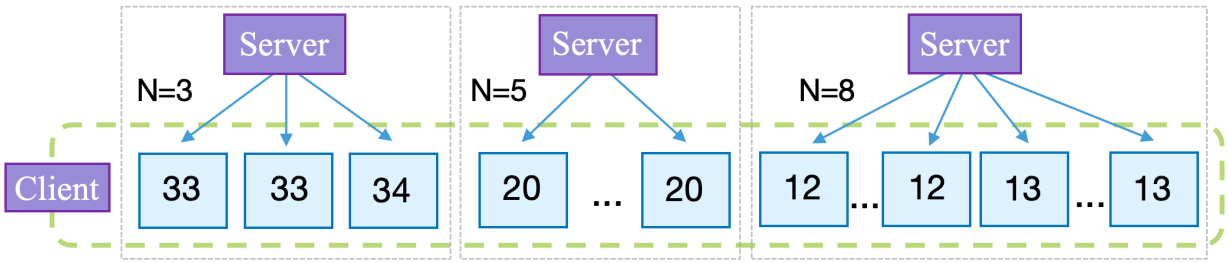


Figure 5: Engine allocation per client across three scenarios, with numbers in blue boxes indicating the number of engines.

3.2. Experiment settings and benchmark model

To illustrate the performance of the AS-EFL-based RUL prognostic method, we consider three categories of benchmark models:

(i) **Baseline model**: This model assumes that data among different clients is independent, with each client conducting model training and test locally. This approach is common in industries [7, 8, 9], while it mitigates the risk of data leakage, the limited quantity and quality of data often lead to suboptimal outcomes.

(ii) **Classical FL model**: This model is adapted from Kamei and Taghipour [16], who applied a classic FL model (based on FedAvg and FedProx) similar to algorithm 1

Table 4: Hyperparameter configurations and FL algorithm combinations.

Algorithm	N	Batch size	Epoch	T	μ
FedAvg	3	128	50	5	-
	5	64	50	8	-
	8	64	50	8	-
FedProx	3	128	80	5	0.01
	5	64	80	5	0.01
	8	64	120	5	0.01

for RUL prediction. Three classic neural networks (DCNN, LSTM, and GRU) are utilized for local client training, and their corresponding FL models are represented as FL-DCNN, FL-LSTM, and FL-GRU, respectively.

(iii) **AS-FL model**: This model is based on the proposed adaptive sampling approach, combined with a classical FL framework. During sampling, the model emphasizes repeatedly iterating over data with lower quality to improve prediction accuracy.

(iv) **EFL model**: This model applies ensemble algorithms from Section 2.2 to the classical FL approach. During each communication round, datasets from all clients are selected for model training and updating on the server. This framework is similar to the methods in [32, 33], but they have not been applied to RUL prediction.

For a fair comparison, all models employ the same network architecture and hyperparameters for model local training. The network architecture and parameter settings of the three local client training networks are as follows: i) The DCNN model consists of four convolutional layers and two fully connected layers, with each convolutional layer configured with 10 output channels. The first three layers employ 10×1 convolutional kernels to capture global features, while the fourth layer utilizes 3×1 convolutional kernels to extract local features. The tanh activation function is applied after all convolutional layers and the first fully connected layer. A dropout rate of 0.5 is set between fully connected layers to prevent overfitting. ii) The GRU model is configured with three GRU layers, each followed by a dropout rate of 0.2. The first layer comprises 128 units, the second layer comprises 64 units, and the third layer comprises 32 units. Following data processing, the output passes

through a fully connected layer with $ReLU$ activation function. iii) The LSTM model is similar to the GRU model, with GRU units replaced by LSTM units. Training parameters for these three models, including batch size and number of epochs, are detailed in Table 4. All models utilize a uniform learning rate of 0.001 for parameter estimation until the specified number of epochs is reached. The reported experimental results are averaged by 5 trials to reduce the effect of randomness. All the experiments are conducted on a computer with a 2.10 GHz Intel Xeon Gold 5318Y CPU, 14 GB RAM, and NVIDIA A16 GPU, utilizing the PyTorch and Scikit-learn libraries in Python.

3.3. Implementation process of proposed model

First, we begin by outlining the implementation process of the proposed model based on Algorithm 2, assuming 5 clients and using the FedProx algorithm for the FL method. The local model consists of three sub-models: LSTM, GRU, and DCNN. For the pre-processed CM data from each client, we train the model using the proposed AS-EFL approach. Table 5 illustrates the dynamic variations in sampling probabilities and model weights during adaptive sampling and ensemble learning. Specifically, in the first round, the sampling probabilities for all clients are set to 1. After performing ensemble learning locally, each client calculates the Score and weight of its models (e.g., in Round 1, the weights of the three base models for Client 1 are 0.127, 0.124, and 0.749). The model parameters are then transmitted to the server. After aggregation, new model parameters $\boldsymbol{\theta}_m^1$ for $m = 1, 2, 3$ are obtained and distributed to each client. The performance metrics of the different models are computed using the new model parameters and sent back to the server. The server then calculates the sampling probabilities for the next round (e.g., in Round 1, the p_k^1 values for 5 clients using the LSTM model are: 0.0042, 0.0147, 0.0001, 0.9805, and 0.0006). In Round 2, three clients are selected based on the probabilities p_k^1 (e.g., Clients 1, 4, and 5 are selected for the LSTM model), and only these clients undergo local ensemble learning training, followed by similar steps as in the previous round. This process continues until the number of training rounds reaches the threshold (e.g., Round 8). Then, based on the trained model, we perform RUL prediction on the CM data from the test set within each respective local client.

Table 5: Probabilities and model weights per round in adaptive sampling and ensemble learning, where (✓) indicates the selected client for training, (×) indicates the client is not selected.

Epoch		Client 1	Client 2	Client 3	Client 4	Client 5
T_1	LSTM	0.127(✓)	0.122(✓)	0.151(✓)	0.113(✓)	0.136(✓)
	GRU	0.124(✓)	0.119(✓)	0.147(✓)	0.110(✓)	0.133(✓)
	DCNN	0.749(✓)	0.759(✓)	0.702(✓)	0.777(✓)	0.731(✓)
T_2	LSTM	0.325(✓)	0.278(×)	0.427(×)	0.351(✓)	0.298(✓)
	GRU	0.361(✓)	0.321(✓)	0.498(×)	0.410(✓)	0.298(×)
	DCNN	0.314(✓)	0.401(×)	0.075(×)	0.239(✓)	0.365(✓)
⋮	⋮	⋮	⋮	⋮	⋮	⋮
T_8	LSTM	0.410(×)	0.331(✓)	0.244(×)	0.215(✓)	0.363(✓)
	GRU	0.200(✓)	0.392(×)	0.344(✓)	0.393(×)	0.261(✓)
	DCNN	0.391(✓)	0.277(×)	0.412(✓)	0.392(×)	0.377(✓)

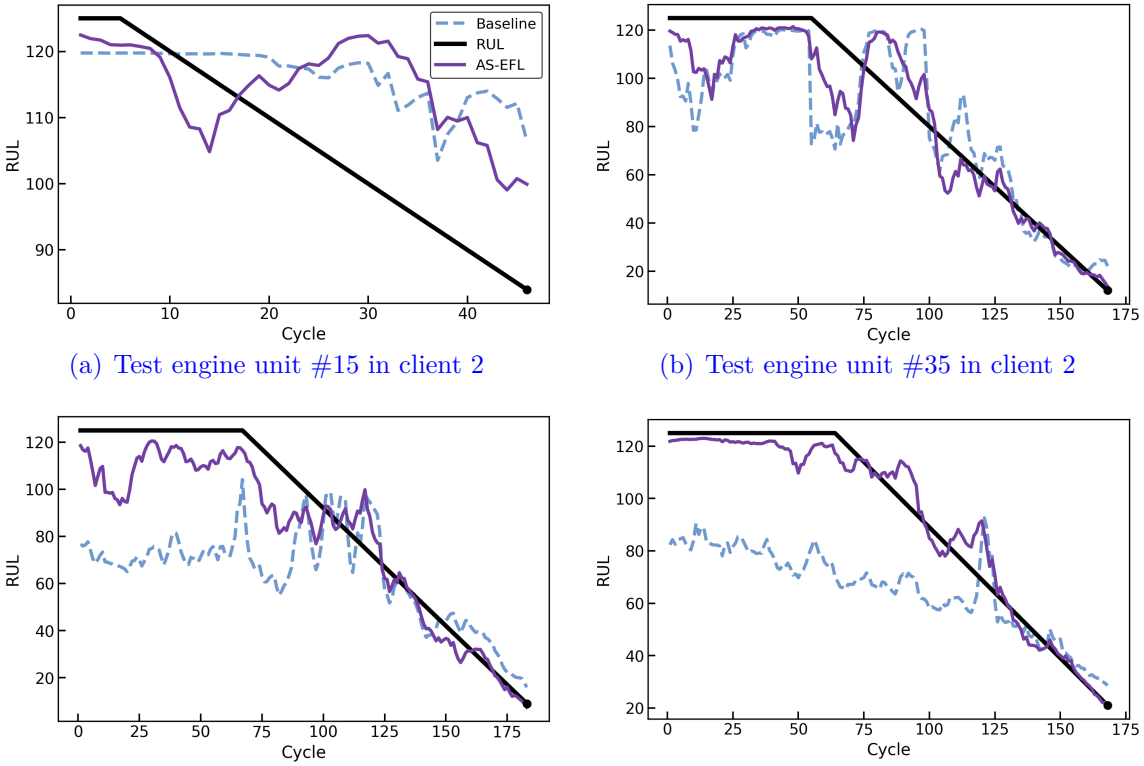
3.4. Prognostic performance comparison

3.4.1. Prognostic performance of proposed model

To evaluate the predictive performance of different models, we use Score and RMSE as evaluation metrics [45, 46]. Their calculation formulas are similar to (6) and (11), with $d_{m,i}$ replaced by $\tilde{d}_{i,k} = \hat{y}_{i,k} - y_{i,k}$ and $s_{m,i}$ replaced by $\tilde{s}_{i,k}$, applied across all clients. Here, $\hat{y}_{i,k}$ represents the predicted RUL value for the i-th system under k-th client. The specific formulas are as follows:

$$\widetilde{\mathrm{SC}} = \sum_{k=1}^{N}\sum_{i=1}^{n_k} \tilde{s}_{i,k}, \quad \text{and} \quad \widetilde{\mathrm{RMSE}} = \sqrt{\frac{\sum_{k=1}^{N}\sum_{i=1}^{n_k} \tilde{d}_{i,k}^2}{\sum_{k=1}^{N} n_k}}. \tag{12}$$

First, we present the predictive trajectory performance of the proposed model for each engine across different numbers of clients. For Classical FL and Baseline models, we utilize the LSTM model, whereas for the proposed models, we employ three distinct model predictors ($M = 3$) for local client training. Due to space constraints and similar conclusions, we randomly select test engine units from different clients as examples. Figure 6 illustrates the RUL prediction trajectories of these four engines under the FedProx algorithm. Compared to the Baseline model, the proposed AS-EFL model demonstrates better prediction accuracy for these engines, particularly in regions with lower RUL values. This indicates that as CM data is continuously acquired and as engine units approach failure, fault characteristics



(a) Test engine unit #15 in client 2

(b) Test engine unit #35 in client 2

(c) Test engine unit #81 in client 3

(d) Test engine unit #100 in client 5

Figure 6: RUL estimation for four test engine units based on five clients in the FedProx algorithm.

become more pronounced and are effectively captured by the model to enhance prediction accuracy. Despite some prediction biases, the model performs particularly well in predicting accurately when the engine units are nearing failure.

Additionally, we provide the RUL prediction results for all test engines under their respective last recorded cycle, based on five clients in the FedProx algorithm, as shown in Figure 7. From the graph, it is evident that the performance of the proposed model is significantly better than that of the Baseline model. The average RMSE for each client is as follows: for the proposed model, it is (2.06, 2.98, 3.42, 2.35), while for the baseline model, it is (4.66, 12.90, 8.40, 5.47). The poor performance of the Baseline method across all test clients exposes the limitation of relying on independently created prediction models by clients, mainly attributed to the relatively limited amount of data available from each client, resulting in limited predictive outcomes for RUL prediction of individual test engines. On the other hand, the proposed AS-EFL model, while ensuring data privacy, can combine information from other clients to provide more accurate RUL prediction results. Note that

the graph also shows instances of poor predictive performance for certain engines (e.g., units #15 in Figure 6(a), #27, #45, and #95). This is mainly due to the short recorded cycles of these engines, which provide limited degradation trend information, resulting in larger prediction errors. We expect that as the operating cycles increase, the predictive performance of the proposed model for these engines will improve, as shown in Figure 6(b)-6(d).

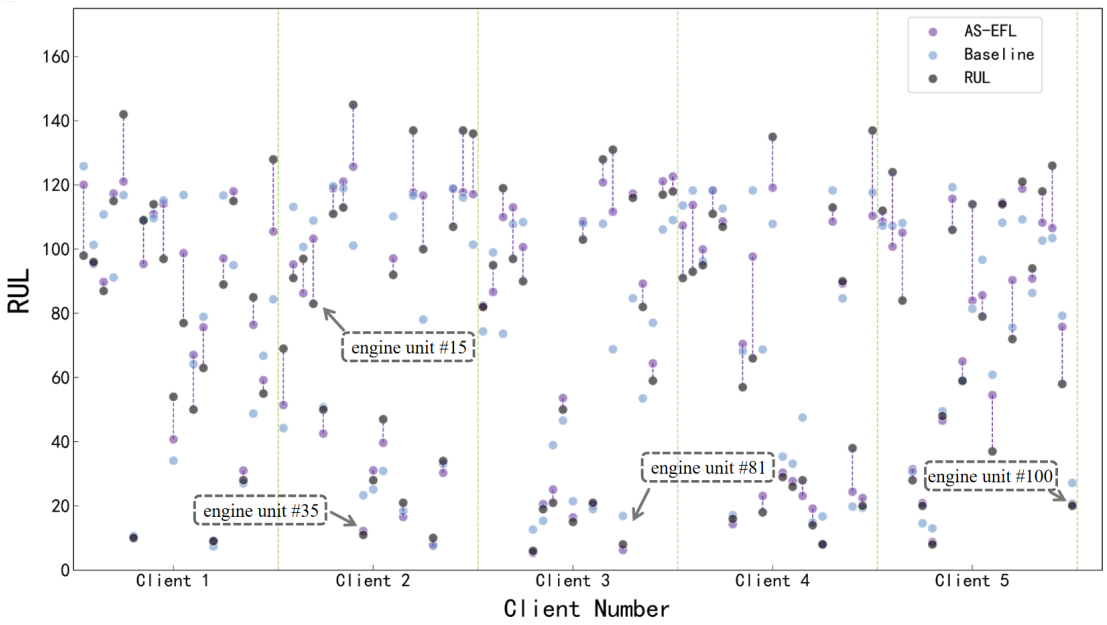


Figure 7: RUL prediction results for all test engines under their respective last recorded cycle.

3.4.2. Comparison with other methods

In this section, we compare the proposed AS-EFL model with other state-of-the-art methods, as summarized in Table 6. Compared to centralized learning, FL offers the advantage of privacy protection, as only model updates are shared, not the data. Despite this, AS-EFL achieves comparable predictive performance to traditional centralized models like MCLSTM and Auto-Encoder, demonstrating that it can maintain high accuracy while protecting privacy. Compared to other FL models, the AS-EFL model significantly outperforms Fed-LSTM [16], emphasizing the advantages of combining AS and ensemble learning.

3.5. Ablation study

Section 3.5.1 evaluates the performance of various models under different client numbers and algorithms. Two ablation studies in Sections 3.5.2 and 3.5.3 further assess the impact

Table 6: Comparison of evaluation metrics with other methods.

Pattern	Method	RMSE	Score
Centralized	Auto-Encoder [10]	13.58	**228.00**
	MCLSTM [11]	13.70	315.00
	ANN-EA [12]	14.24	-
	GA-RBM-LSTM [13]	12.60	273.70
	BiLSTM-ED [14]	14.70	273.00
	GHDR [15]	**11.58**	281.65
Decentralized	Fed-LSTM [16]	15.24	304.80
	AS-EFL	13.25	265.81

of individual modules on the proposed model's performance.

3.5.1. Overall performance evaluation

First, we present the average predictive performance results of different models under varying numbers of clients, as shown in Table 7. It can be observed that AS-EFL consistently achieves the lowest RMSE and score across all comparisons. Compared to traditional FL methods, AS-EFL reduces RMSE by an average of approximately 15.6% and the score by 48.6%, highlighting its advantages in optimizing model performance. Moreover, integrating individual modules into FL methods can enhance predictive performance. For instance, AS-FL reduces RMSE by approximately 11.3% and the score by 32.3%, while EFL reduces RMSE by 12.8% and the score by 39.8%.

3.5.2. Improvement through ensemble learning on FL framework

We analyze the impact of ensemble learning on RUL prediction within the FL framework for both $M = 2$ and $M = 3$ scenarios. In the $M = 2$ setting, three combinations are considered: a) LSTM with GRU, b) LSTM with DCNN, and c) GRU with DCNN. For $M = 3$, the combination includes LSTM, GRU, and DCNN. Figure 8 illustrates the RUL prediction results for different EFL model combinations using the FedProx algorithm. We can see that in experiments with 8 clients, the GRU and DCNN combination slightly outperforms the $M = 3$ model in RMSE. However, the $M = 3$ model consistently achieves higher predictive accuracy in other settings. Notably, any two-model combination surpasses

Table 7: Average predictive performance results of different models under varying numbers of clients.

Algorithm	Model	RMSE			Score		
		$N = 3$	$N = 5$	$N = 8$	$N = 3$	$N = 5$	$N = 8$
	Baseline	19.53	19.11	20.61	1112.36	967.38	1053.45
FedAvg	FL	16.35	16.59	16.52	526.09	509.09	523.58
	AS-FL	14.79	14.59	15.36	365.35	343.81	392.42
	EFL	14.34	14.68	14.48	**276.98**	309.14	348.52
	AS-EFL	**13.87**	**13.25**	**13.67**	278.63	**265.81**	**293.75**
FedProx	FL	16.27	16.46	16.79	533.61	492.13	499.22
	AS-FL	14.72	14.95	15.62	351.66	359.05	419.16
	EFL	14.27	14.89	14.79	314.68	331.27	349.03
	AS-EFL	**13.29**	**13.62**	**13.95**	**283.52**	**279.48**	**292.13**

classical FL models with a single model (e.g., FL-DCNN, FL-LSTM, FL-GRU), highlighting ensemble learning's ability to integrate features from multiple models, mitigate individual predictive limitations.

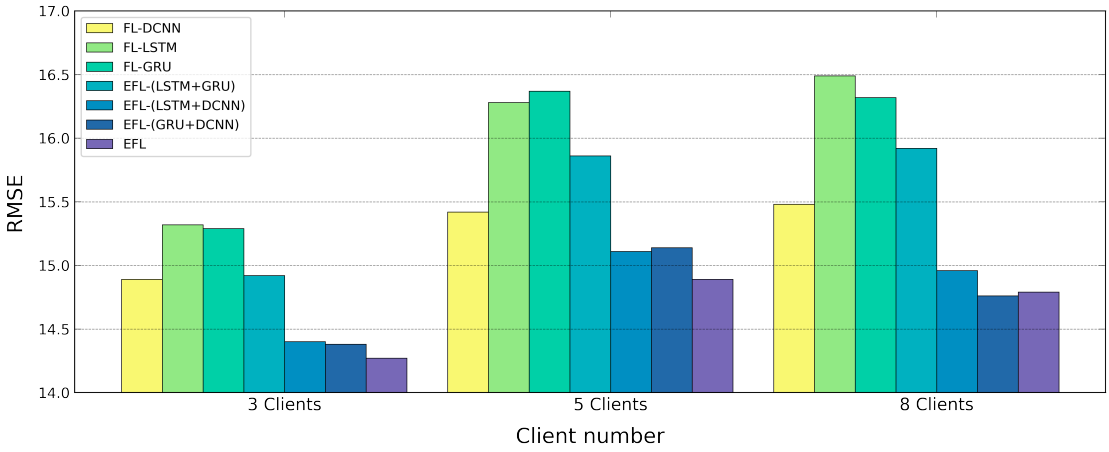
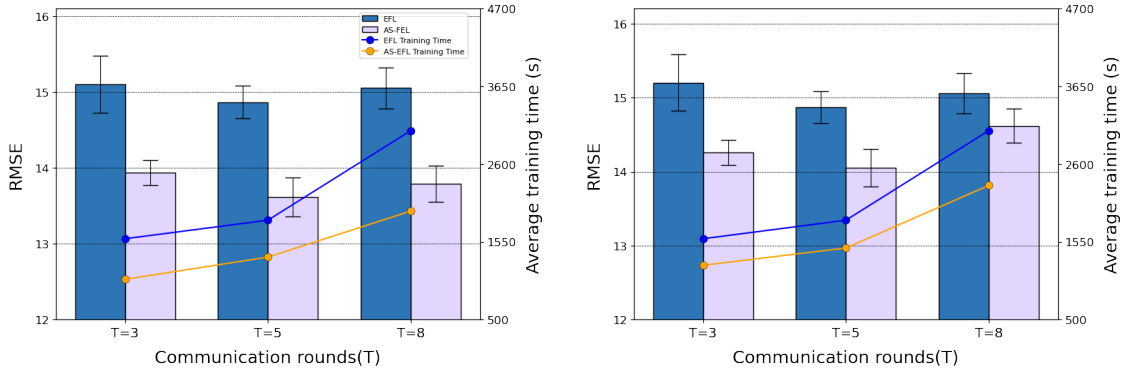


Figure 8: RUL prediction results based on different combinations of EFL model.

3.5.3. Improvement through adaptive sampling on EFL framework

We evaluate the impact of different communication rounds T and the proportion of selected clients $q_t = \mathcal{S}_t/N$ on RUL prediction accuracy, as well as the corresponding error

distribution (see Figure 9). Additionally, we calculate the total training time for all clients under the FL framework. The results show that the AS-EFL model consistently outperforms the EFL model across all experiments, with particularly significant improvements observed at $q_t = 60\%$. Under the same q_t, $T = 5$ achieves better predictive accuracy, while at the same T, $q_t = 60\%$ performs better than $q_t = 80\%$. The slight errors observed can be attributed to model design factors, such as parameter changes across training rounds, which introduce minor deviations. However, these errors remain within an acceptable range, and the overall performance of the model is stable. Beyond accuracy improvements, the AS-EFL model also significantly reduces training time, achieving a reduction of approximately 20% to 40% compared to the EFL model. This efficiency gain is primarily attributed to the AS training module, which optimizes client selection strategies, reducing unnecessary computations and accelerating the model's progression. The impact of more extreme sampling frequencies on prediction performance can be found in Supplementary Section S2.



(a) FD001 with $q_t = 60\%$ (b) FD001 with $q_t = 80\%$

Figure 9: Impact of different T and q_t values on RMSE and training time in the FedProx algorithm when $N = 5$.

3.6. Sensitivity analysis

This section extends the proposed model to more practical scenarios, such as data imbalance (see Section 3.6.1), missing data (see Section 3.6.2), and scale variation (see Section 3.6.3), evaluating the model's estimation performance and efficiency. Additionally, we provide performance results for long-term predictions (see Supplementary Section S3).

3.6.1. Effect of data imbalance

In industry, data imbalance often arises from varying data volumes across clients. For instance, one client may have data from 10 engines, while another has data from 50. This imbalance can affect FL model training, as clients with larger data volumes may dominate the global model, neglecting smaller clients' contributions. This issue is common in fields such as manufacturing process optimization and medical data analysis. To address this, we design an imbalanced data partitioning scheme to evaluate the model's performance. For $N = 5$, we sample two data partitioning schemes: (a) Scheme 1: Five clients with data from 10, 15, 20, 25, and 30 engines. (b) Scheme 2: Five clients with data from 5, 10, 15, 20, and 50 engines.

Table 8 shows the RUL prediction results of the proposed model under two schemes. The results indicate that AS-EFL consistently achieves better accuracy. For example, in Scheme 2, AS-EFL's RMSE is 14.4% lower and Score 42.0% lower than the traditional FL model, demonstrating better adaptation to data distribution differences. The main reason is that traditional FL struggles with controlling client participation, allowing clients with larger datasets to dominate the global model, while clients with smaller datasets are often overlooked, potentially reducing prediction accuracy. In contrast, the AS module dynamically focuses on clients with poorer data quality, ensuring their contributions are effectively utilized in training, while the EFL module integrates features from multiple global models, further enhancing the model's generalization and robustness.

Table 8: Comparison of RUL prediction with other models under imbalanced data.

Algorithm	Model	RMSE		Score	
		Scheme 1	Scheme 2	Scheme 1	Scheme 2
	Baseline	20.93	19.78	1162.75	1054.32
FedAvg	FL	16.70	16.12	479.93	468.12
	EFL	14.83	14.48	336.78	315.06
	AS-EFL	**13.91**	**13.89**	**303.39**	**287.15**
FedProx	FL	16.63	16.16	521.65	513.33
	EFL	14.42	14.26	323.24	291.05
	AS-EFL	**13.80**	**13.73**	**308.15**	**280.18**

3.6.2. Effect of missing data

Data missing is a common issue in industrial applications, often caused by sensor failures or network problems. To simulate this, we randomly deleted a percentage of data (including both training and testing sets) in rows (cycles) to evaluate the model's performance under missing data conditions. Table 9 shows the impact of different data missing percentages on Client 1's performance under the FedAvg framework ($N = 5$). The results show that with low to moderate data missing (e.g., 10% and 15%), the proposed model maintains good accuracy. For instance, with 10% data missing, RMSE increases by only 5.2%, and the Score shows almost no significant change. This is mainly due to the collaboration of ensemble learning and adaptive sampling, which helps mitigate the negative impact of missing data. However, when data missing increases to 25% or 30%, performance degradation becomes more noticeable. For example, with 30% data missing, RMSE rises to 21.26, and the Score increases to 143.58, due to reduced training data and disrupted time-series integrity.

To further address the impact of high data missing, we introduced two compensation mechanisms: mean imputation and linear interpolation. With 30% missing data, mean imputation resulted in RMSE of 17.49 and Score of 87.06, while linear interpolation gave RMSE of 18.31 and Score of 94.27. Compared to no imputation (RMSE = 21.26, Score = 143.58), mean imputation reduced RMSE by 17.7% and Score by 39.4%, while linear interpolation reduced RMSE by 13.9% and Score by 34.3%. These results show that imputation methods can significantly improve model performance in the presence of missing data.

Table 9: The impact of different data missing percentages on the prediction performance of Client 1.

Matrix	Missing percentages						
	0%	5%	10%	15%	20%	25%	30%
RMSE	16.67	17.28	17.54	18.67	18.87	20.70	21.26
Score	82.38	86.03	85.23	100.16	104.38	119.24	143.58

3.6.3. Effect of data scale

We analyze the proposed model's scalability and resource requirements by varying the number of clients. The data is divided into five clients, each containing 20 engines. Training begins with one client and gradually increases to all clients, to assess the impact of client and data size on prediction performance, and communication overhead. Table 10 shows the

prediction performance and time overhead (average training, prediction, and communication time) of client 1 under the FedAvg framework. The results are as follows: (a) As the number of clients increases, prediction performance improves, indicating that more clients enhance accuracy. (b) The training time per round remains consistent, ranging from 27 to 29 seconds, suggesting that increasing the number of clients does not significantly affect the training overhead for individual clients. (c) The prediction time is notably short, which meets the demands of real-time RUL prediction, allowing for quick and efficient forecasting. (d) Communication time is minimal, with a slight increase as the number of clients grows, from 0.0018 seconds to 0.0125 seconds.

Table 10: Prediction performance and time overhead of client 1 under different data scales.

Number of clients	RMSE	Score	Time (s)		
			Training	Prediction	Communication
1	20.68	145.26	29.38	**0.1489**	**0.0018**
2	19.10	123.88	**27.17**	0.1583	0.0027
3	17.46	101.81	27.76	0.1576	0.0086
4	17.65	93.49	27.56	0.1618	0.0113
5	**16.67**	**82.38**	27.64	0.1602	0.0125

4. Conclusion

This study introduces an AS-based ensemble FL method for RUL prediction, tackling challenges like limited sample sizes and data privacy in modern industrial environments. Compared to traditional FL approaches, the proposed method offers several advantages: i) Ensemble learning aggregates predictions from multiple models, improving accuracy and generalization. ii) The AS-based approach dynamically adjusts data collection strategies based on client data quality. We apply the method to a turbofan engine dataset and compare it with benchmark strategies. Key findings include: (a) The AS-EFL method significantly outperforms independent client learning models and achieves accuracy close to state-of-the-art centralized methods. (b) The ablation study shows that integrating individual modules into FL improves predictive performance, with RMSE reducing by 12% and Score by 35%

on average. (c) Sensitivity analysis confirms that the proposed model excels in prediction performance and efficiency, even under conditions of data imbalance, missing data, and scale variation.

These results highlight the effectiveness of the AS-EFL method. However, several areas for future improvement are identified: (a) Although the model shows good prediction speed, it is designed for batch processing and lacks real-time training or updates. Future work will focus on online learning architectures to handle continuous data and adapt to changing conditions. (b) Although validated on the C-MAPSS dataset and extended to realistic scenarios such as missing data and client data imbalance, this single-source dataset may not fully capture the complexity of real-world industrial settings. Future research will use more dynamic datasets and varied industrial scenarios to test the model's generalization. (c) Traditional imputation methods may not effectively handle higher missing rates or complex missing patterns. Future work will explore advanced techniques, such as using generative adversarial networks to generate missing data or applying data augmentation strategies. (d) Although we have conducted experiments on scale variation, the current scope remains limited and has not yet exceeded expectations. Future efforts will focus on expanding the scale by increasing the number of clients and incorporating larger datasets for a more comprehensive evaluation.

Acknowledgment

The research is supported by the project of Economic Forecasting and Policy Simulation Laboratory, Zhejiang Gongshang University (No. 2024SYS019), the Fundamental Research Funds for the Provincial Universities of Zhejiang, the Summit Advancement Disciplines of Zhejiang Province (Zhejiang Gongshang University—Statistics), and Collaborative Innovation Center of Statistical Data Engineering Technology & Application.

References

[1] A. Thelen, M. Li, C. Hu, E. Bekyarova, S. Kalinin, M. Sanghadasa, Augmented model-based framework for battery remaining useful life prediction, Applied Energy 324 (2022) 119624. doi:10.1016/j.apenergy.2022.119624.

[2] Q. Zhai, Z.-S. Ye, RUL prediction of deteriorating products using an adaptive Wiener process model, IEEE Transactions on Industrial Informatics 13 (2017) 2911–2921. doi:10.1109/TII.2017.2684821.

[3] J. Guo, Z. Li, M. Li, A review on prognostics methods for engineering systems, IEEE Transactions on Reliability 69 (2019) 1110–1129. doi:10.1109/TR.2019.2957965.

[4] M. Kordestani, M. Saif, M. E. Orchard, R. Razavi-Far, K. Khorasani, Failure prognosis and applications—A survey of recent literature, IEEE Transactions on Reliability 70 (2019) 728–748. doi:10.1109/TR.2019.2930195.

[5] V. Nemani, L. Biggio, X. Huan, Z. Hu, O. Fink, A. Tran, Y. Wang, X. Zhang, C. Hu, Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial, Mechanical Systems and Signal Processing 205 (2023) 110796. doi:10.1016/j.ymssp.2023.110796.

[6] L. Ren, Z. Jia, Y. Laili, D. Huang, Deep learning for time-series prediction in IIoT: progress, challenges, and prospects, IEEE Transactions on Neural Networks and Learning Systems 35 (2023) 15072–15091. doi:10.1109/TNNLS.2023.3291371.

[7] M. Soualhi, K. T. P. Nguyen, K. Medjaher, F. Nejjari, V. Puig, J. Blesa, J. Quevedo, F. Marlasca, Dealing with prognostics uncertainties: Combination of direct and recursive remaining useful life estimations, Computers in Industry 144 (2023) 103766. doi:10.1016/j.compind.2022.103766.

[8] W. Peng, Z.-S. Ye, N. Chen, Bayesian deep-learning-based health prognostics toward prognostics uncertainty, IEEE Transactions on Industrial Electronics 67 (2019) 2283–2293. doi:10.1109/TIE.2019.2907440.

[9] Q. Xu, M. Wu, E. Khoo, Z. Chen, X. Li, A hybrid ensemble deep learning approach for early prediction of battery remaining useful life, IEEE/CAA Journal of Automatica Sinica 10 (2023) 177–187. doi:10.1109/JAS.2023.123024.

[10] W. Yu, I. Y. Kim, C. Mechefeske, An improved similarity-based prognostic algorithm for rul estimation using an rnn autoencoder scheme, Reliability Engineering & System Safety 200 (2020) 106926. doi:10.1016/j.ress.2020.106926.

[11] S. Xiang, Y. Qin, J. Luo, H. Pu, B. Tang, Multicellular lstm-based deep learning model for aero-engine remaining useful life prediction, Reliability Engineering and System Safety 216 (2021) 107927. doi:10.1016/j.ress.2021.107927.

[12] D. Laredo, Z. Chen, O. Schütze, J.-Q. Sun, A neural network-evolutionary computational framework for remaining useful life estimation of mechanical systems, Neural Networks 116 (2019) 178–187. doi:10.1016/j.neunet.2019.04.016.

[13] W. Yu, I. Y. Kim, C. Mechefeske, Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme, Mechanical Systems and Signal Processing 129 (2019) 764–780. doi:10.1016/j.ymssp.2019.05.005.

[14] A. L. Ellefsen, E. Bjørlykhaug, V. Åsøy, S. Ushakov, H. Zhang, Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture, Reliability Engineering & System Safety 183 (2019) 240–251. doi:10.1016/j.ress.2018.11.027.

[15] X. Chen, H. Wang, S. Lu, J. Xu, R. Yan, Remaining useful life prediction of turbofan engine using global health degradation representation in federated learning, Reliability Engineering & System Safety 239 (2023) 109511. doi:10.1016/j.ress.2023.109511.

[16] S. Kamei, S. Taghipour, A comparison study of centralized and decentralized federated learning approaches utilizing the transformer architecture for estimating remaining useful life, Reliability Engineering & System Safety 233 (2023) 109130. doi:10.1016/j.ress.2023.109130.

[17] D. B. Stringer, P. N. Sheth, P. E. Allaire, Physics-based modeling strategies for diagnostic and prog-

nostic application in aerospace systems, Journal of Intelligent Manufacturing 23 (2012) 155–162.

[18] L. Zhuang, A. Xu, Y. Wang, Y. Tang, Remaining useful life prediction for two-phase degradation model based on reparameterized inverse Gaussian process, European Journal of Operational Research 319 (2024) 877–890. doi:10.1016/j.ejor.2024.06.032.

[19] J. D. Fernández, S. P. Menci, C. M. Lee, A. Rieger, G. Fridgen, Privacy-preserving federated learning for residential short-term load forecasting, Applied energy 326 (2022) 119915. doi:10.1016/j.apenergy.2022.119915.

[20] L. Li, Y. Fan, M. Tse, K.-Y. Lin, A review of applications in federated learning, Computers & Industrial Engineering 149 (2020) 106854. doi:10.1016/j.cie.2020.106854.

[21] I. Kevin, K. Wang, X. Zhou, W. Liang, Z. Yan, J. She, Federated transfer learning based cross-domain prediction for smart manufacturing, IEEE Transactions on Industrial Informatics 18 (2021) 4088–4096. doi:10.1109/TII.2021.3088057.

[22] T. Kröger, A. Belnarsch, P. Bilfinger, W. Ratzke, M. Lienkamp, Collaborative training of deep neural networks for the lithium-ion battery aging prediction with federated learning, eTransportation 18 (2023) 100294. doi:10.1016/j.etran.2023.100294.

[23] X. Chen, H. Wang, S. Lu, R. Yan, Bearing remaining useful life prediction using federated learning with Taylor-expansion network pruning, IEEE Transactions on Instrumentation and Measurement 72 (2023) 1–10. doi:10.1109/tim.2023.3264027.

[24] D. Zhang, W. Tian, X. Cheng, F. Shi, H. Qiu, X. Liu, S. Chen, FedBIP: A federated learning-based model for wind turbine blade icing prediction, IEEE Transactions on Instrumentation and Measurement 72 (2023) 1–11. doi:10.1109/tim.2023.3273675.

[25] L. Guo, Y. Yu, M. Qian, R. Zhang, H. Gao, Z. Cheng, FedRUL: A new federated learning method for edge-cloud collaboration based remaining useful life prediction of machines, IEEE/ASME Transactions on Mechatronics 28 (2023) 350–359. doi:10.1109/tmech.2022.3195524.

[26] N. H. Du, N. H. Long, K. N. Ha, N. V. Hoang, T. T. Huong, K. P. Tran, Trans-Lighter: A light-weight federated learning-based architecture for remaining useful lifetime prediction, Computers in Industry 148 (2023) 103888. doi:10.1016/j.compind.2023.103888.

[27] J. Zhang, J. Tian, P. Yan, S. Wu, H. Luo, S. Yin, Multi-hop graph pooling adversarial network for cross-domain remaining useful life prediction: A distributed federated learning perspective, Reliability Engineering & System Safety 244 (2024) 109950. doi:10.1016/j.ress.2024.109950.

[28] L. Wang, Z. Zhu, X. Zhao, Dynamic predictive maintenance strategy for system remaining useful life prediction via deep learning ensemble method, Reliability Engineering & System Safety 245 (2024) 110012. doi:10.1016/j.ress.2024.110012.

[29] Y. Li, J. Li, H. Wang, C. Liu, J. Tan, Knowledge enhanced ensemble method for remaining useful life prediction under variable working conditions, Reliability Engineering & System Safety 242 (2024) 109748. doi:10.1016/j.ress.2023.109748.

[30] Z. Mian, X. Deng, X. Dong, Y. Tian, T. Cao, K. Chen, T. Al Jaber, A literature review of fault diagnosis based on ensemble learning, Engineering Applications of Artificial Intelligence 127 (2024) 107357. doi:10.1016/j.engappai.2023.107357.

[31] Y. Cheng, J. Zeng, Z. Wang, D. Song, A health state-related ensemble deep learning method for aircraft

engine remaining useful life prediction, Applied Soft Computing 135 (2023) 110041. doi:10.1016/j.asoc.2023.110041.

[32] J. Wang, J. Hu, J. Mills, G. Min, M. Xia, N. Georgalas, Federated ensemble model-based reinforcement learning in edge computing, IEEE Transactions on Parallel and Distributed Systems 34 (2023) 1848–1859.

[33] N. Shi, F. Lai, R. Al Kontar, M. Chowdhury, Fed-ensemble: Ensemble models in federated learning for improved generalization and uncertainty quantification, IEEE Transactions on Automation Science and Engineering 21 (2023) 2792–2803. doi:10.1109/tase.2023.3269639.

[34] R. Kontar, N. Shi, X. Yue, S. Chung, E. Byon, M. Chowdhury, J. Jin, The internet of federated things (IoFT), IEEE Access 9 (2023) 156071–156113. doi:10.1109/ACCESS.2021.3127448.

[35] A. Mabrouk, R. P. Díaz Redondo, M. Abd Elaziz, M. Kayed, Ensemble federated learning: An approach for collaborative pneumonia diagnosis, Applied Soft Computing 144 (2023) 110500. doi:10.1016/j.asoc.2023.110500.

[36] M. M. Alam, T. Ahmed, M. Hossain, M. H. Emo, M. K. I. Bidhan, M. T. Reza, M. G. R. Alam, M. M. Hassan, F. Pupo, G. Fortino, Federated ensemble-learning for transport mode detection in vehicular edge network, Future Generation Computer Systems 149 (2023) 89–104. doi:10.1016/j.future.2023.07.022.

[37] C. Zhang, Y. Ma, Ensemble machine learning: methods and applications, Springer, 2012.

[38] X. Li, Q. Ding, J.-Q. Sun, Remaining useful life estimation in prognostics using deep convolutional neural networks, Reliability Engineering & System Safety 172 (2018) 1–11. doi:10.1016/j.ress.2017.11.021.

[39] J. Zhu, N. Chen, W. Peng, Estimation of bearing remaining useful life based on multiscale convolutional neural network, IEEE Transactions on Industrial Electronics 66 (2018) 3208–3216. doi:10.1109/TIE.2018.2844856.

[40] J. Wang, J. Yan, C. Li, R. X. Gao, R. Zhao, Deep heterogeneous GRU model for predictive analytics in smart manufacturing: Application to tool wear prediction, Computers in Industry 111 (2019) 1–14. doi:10.1016/j.compind.2019.06.001.

[41] Z. Zhao, J. Wu, F. Cai, S. Zhang, Y.-G. Wang, A hybrid deep learning framework for air quality prediction with spatial autocorrelation during the COVID-19 pandemic, Scientific Reports 13 (2023) 1015. doi:10.1038/s41598-023-28287-8.

[42] A. Saxena, K. Goebel, D. Simon, N. Eklund, Damage propagation modeling for aircraft engine run-to-failure simulation, in: 2008 international conference on prognostics and health management, 2008, pp. 1–9. doi:10.1109/PHM.2008.4711414.

[43] M. Kim, K. Liu, A Bayesian deep learning framework for interval estimation of remaining useful life in complex systems by incorporating general degradation characteristics, IISE Transactions 53 (2020) 326–340. doi:10.1080/24725854.2020.1766729.

[44] L. Zhuang, A. Xu, X.-L. Wang, A prognostic driven predictive maintenance framework based on Bayesian deep learning, Reliability Engineering & System Safety 234 (2023) 109181. doi:10.1016/j.ress.2023.109181.

[45] Q. Zhai, Y. Li, P. Chen, Modeling product degradation with heterogeneity: A general random-effects

Wiener process approach, IISE Transactions (2024). doi:10.1080/24725854.2024.2434125.

[46] M. L. H. Souza, C. A. da Costa, G. de Oliveira Ramos, A machine-learning based data-oriented pipeline for prognosis and health management systems, Computers in Industry 148 (2023) 103903. doi:10.1016/j.compind.2023.103903.