JOURNAL OF MACHINE LEARNING AND APPLICATIONS
Jan-Mar 2025, VOL. 1, ISSUE 1, pp. 12-14
https://doi.org/10.61577/jmla.2025.100005

RESEAPRO JOURNALS

NARRATIVES

OPEN ACCESS

Using federated machine learning in predictive maintenance of jet engines

Asaph Matheus Barbosa[1*], Thao Vy Nhat Ngo[1], Elaheh Jafarigol[1], Theodore B. Trafalis[1], Emuobosa P. Ojoboh[1]

Department of Data Science, University of Oklahoma, Norman, USA

ABSTRACT

The aim of this research is to predict the Remaining Useful Life (RUL) of turbine jet engines using a federated machine learning framework, ensuring data privacy and security while maintaining high predictive accuracy. Federated Learning (FL) enables multiple edge devices/nodes or servers to collaboratively train a shared model without sharing sensitive data, making it ideal for industries like aviation, where data confidentiality is paramount. The proposed system employs Long Short-Term Memory (LSTM) networks, a type of recurrent neural network, to model the complex temporal relationships and degradation patterns in engine data. By leveraging decentralized computation, the framework allows models to be trained locally on each device, with learned weights aggregated at a central server using the Federated Averaging (FedAvg) algorithm. The study utilizes the C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) dataset, a publicly available resource from NASA, which simulates engine degradation under various operational conditions. The dataset includes time-series data from 21 sensors and three operational settings, providing a comprehensive foundation for analyzing fault progression and failure modes. Computational results demonstrate the effectiveness of the proposed approach; with validation and test Root Mean Squared Error (RMSE) metrics range from 13.811 to 22.998 across different operational scenarios. The aggregated FL model achieved an RMSE of 17.899, showcasing its ability to generalize across diverse conditions. By accurately predicting the RUL of jet engines, this approach enables optimized maintenance schedules, reduced downtime, and improved operational efficiency, ultimately leading to cost savings and enhanced performance in the aviation industry. This research highlights the advantages of federated learning in handling sensitive data while achieving robust predictive performance for critical industrial applications.

KEYWORDS

Federated learning; predictive maintenance; Privacy; Model accuracy; Long-short term memory

ARTICLE HISTORY

Received 19 February 2025;
Revised 19 March 2025;
Accepted 27 March 2025

Introduction

Integrating privacy-preserving machine learning (ML) techniques—particularly Federated Learning (FL)—into Predictive Maintenance (PM), represents a significant shift in how industrial operations manage maintenance and data security. This approach has broad implications for operational efficiency, data privacy, regulatory compliance, and technological innovation. In FL, data are processed locally at the device or server level, drastically reducing the risk of exposing sensitive information during transmission or in a centralized database [1]. This is crucial for industries where operational data may include proprietary or sensitive business information. By minimizing data centralization, federated learning decreases the vulnerability of systems to massive data breaches, a growing concern with the increasing incidences of cyber-attacks. This work investigates the application of FL to PM tasks, focusing on the utilization of Long Short-Term Memory (LSTM) networks for predicting engine faults. Our research is motivated by the increasing demand for efficient PM methodologies that can minimize downtime and reduce operational costs in various industries [2]. Privacy-preserving ML enables real-time data analysis directly on the machines where data is generated. This allows for immediate identification of potential issues, facilitating quicker responses to prevent failures. With the ability to analyze data across a network of devices without compromising privacy, organizations can optimize maintenance schedules based on predictive insights, rather than reactive or scheduled maintenance strategies. This not only extends the life of equipment but also reduces unnecessary downtime. FL aligns with global data protection regulations towards treating privacy as a fundamental human right and establishing robust privacy protection mechanisms in the era of artificial intelligence [3,4]. The latest update of the National Artificial Intelligence R&D Strategic Plan by the White House in 2023 underscores the significance of FL in addressing data privacy and security concerns. [1]This plan elaborates on long-term investment strategies in responsible AI research, emphasizing the need for advancements in privacy-preserving data sharing and the ongoing challenges within FL. The General Data Protection Regulation (GDPR)[2] in Europe also emphasizes data minimization, privacy by design, and the principle of processing data close to its source. Since FL does not require data to leave its source, it simplifies compliance with laws that restrict cross-border data transfers, making it an attractive option for multinational corporations. In FL only essential model-related information is transmitted. Therefore, FL can significantly reduce the costs associated

***Correspondence:** Mr. Asaph Matheus Barbosa, Department of Data Science, University of Oklahoma, Norman, USA. e-mail: asaphmatheus.barbosa@gmail.com

with data transmission. In addition, by processing data locally and not requiring a central repository for vast amounts of raw data, companies can save on storage costs.

Through iterative experimentation and parameter tuning, we refine our models' accuracy, utilizing FL to distribute the computational load and enhance data privacy. The results section provides an in-depth analysis of the models' performance. Federated learning allows for the development of highly tailored models that learn from diverse data sources without compromising sensitive information. This capability can drive innovation in PM technologies. Different entities, even competitors, can collaborate to improve predictive models without sharing sensitive data, accelerating industry-wide advancements in maintenance strategies. Managing FL across many devices and locations introduces complexity, especially when coordinating updates and maintaining consistent model performance across diverse environments. Implementing an FL system requires sophisticated infrastructure and a shift in traditional data management strategies, which might be challenging for some organizations. This study encompasses a process validation section, where we underscore our collaboration with industry experts to ensure that our research objectives align with practical applications and that our findings remain relevant. The use of privacy-preserving ML and FL in PM strengthens data security and enhances operational efficiencies and compliance with regulatory norms. These technologies are setting new standards for how industries approach maintenance tasks while safeguarding critical data. As adoption grows, they could redefine best practices for asset management across various sectors, promising a future where PM is both more effective and inherently secure. In summary, this study delves into the application of federated learning for PM, presenting a structured approach to model development employing LSTM networks. Our findings aim to contribute to the ongoing discussion on the potential of FL in industrial applications, particularly in enhancing PM strategies to achieve operational efficiency and reliability.

Definition and Background of the Problem

Aircraft maintenance historically has two main philosophies: reactive and proactive. Both are widely used due to their different advantages and disadvantages. Reactive maintenance describes the process of waiting for the life cycle of a part of an airplane subsystem to completely run out before repairing or replacing the faulty components [5]. Proactive maintenance describes the process of scheduling regular maintenance to repair/replace components before they become faulty [6]. The advantage of reactive maintenance is that we can get 100% usage out of our parts, but the obvious disadvantage is that there is a high chance of component failure happening during flights. This can work for something non-critical like overhead cockpit lights, which would not force a flight to be grounded if they failed in flight. On the other hand, something like a High-Pressure Compressor (HPC) failure in flight could prove disastrous. In these scenarios, it is better to perform proactive maintenance, where the disadvantage is that we lose some usage from our components, but we limit the number of in-flight failures. In more recent years, with the advancements of ML, PM has become more popular as a third approach where we can use machine learning to schedule our maintenance for high-risk systems and still get close to 100% usage with a low-error model [7,8]. There are two main problems with this approach: small fleets with small sample sizes to train their PM models and large fleets unwilling to share their plentiful data with competitors due to privacy concerns. The work done in this study applies to FL to address both problems with a single solution.

Data

NASA Ames Prognostics Center of Excellence (PCoE) researchers conducted engine degradation simulations using the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) [9]. The C-MAPSS dataset is publicly available and readily accessible on the NASA website, which can serve as a valuable resource for studying and analyzing engine degradation behaviors in various operational scenarios. The data was converted to .csv and is stored on our GitHub repository.

The C-MAPSS dataset is an operational behavior dataset from different engines. It offers a detailed look into the normal operational conditions of engines, including the presence of noise. Each data entry in the dataset contains 26 columns, encompassing information such as unit number, time cycles, three operational settings, and 21 sensor measurements (see Appendix A for details). These data snapshots, taken during individual operational cycles, provide valuable insights into the engine's behavior. Table 1 provides detailed information about each node. Sensor measurements are observed to be contaminated with noise, which can potentially introduce inaccuracies and inconsistencies in the data analysis process [10].

The dataset is structured into four training and four testing datasets, each with varying numbers of trajectories, conditions, and fault modes. The training sets are designed to showcase examples of faults that grow in magnitude until system failure occurs, providing valuable learning opportunities for predictive maintenance and fault detection. The testing sets may end before system failure, allowing for the evaluation of predictive models under different scenarios.

Table 1. Trajectories and Conditions for each node

Agent Name	Training Set Size	Testing Set Size	Simulation Condition	Faults
FD001	100	100	1	HPC
FD002	260	259	6	HPC
FD003	100	100	1	Fan/HPC
FD004	248	249	6	Fan/HPC

Heatmaps

Heatmaps are powerful visualization tools used to identify correlations between variables in a dataset, making them particularly useful for analyzing relationships among sensor measurements [11]. By plotting a heat map of the correlation matrix, patterns of correlation (both positive and negative) between pairs of variables can be easily visualized through color gradients. In the plot below, the lightest (white/tan) and darkest (black) colors indicate the highest linear correlation among

https://doi.org/10.61577/jmla.2025.100005

variables. The orange/red color indicates there is little to no linear correlation between the variables.

According to Figure 1 of FD001, the heat-map shows the strongest relationships consisting of SM2, SM3, SM4, SM7, SM8, SM11, SM12, SM13, SM15, SM17, SM20, SM21 are correlated with almost all sensor measurement except SM6, SM9, and SM14. Regarding the trend, it seems like RUL has a similar correlation trend with SM7, SM12, SM20, and SM21, which can imply that RUL is mainly determined by SM7, SM12, SM20, and SM21. The other interesting part is that OS1 and OS2 do not have any relationship with the sensor measurements. In FD002, the RUL has no relationship with any of the sensor measurements. SM15 has the strongest relationship with every other sensor measurement, but it seems like it has the opposite trend compared to others. Every operational setting and sensor measurements are correlated with each other except for the SM13 and SM14. Unlike FD001, FD002 shows that OS1 and OS2 are affected by most of the sensor measurements. In FD003, the heatmaps show similar trends and relationships among the operational setting and sensor measurements as in FD001. For FD004, the heatmaps show similar trends and relationships among the operational setting and sensor measurements as FD002.

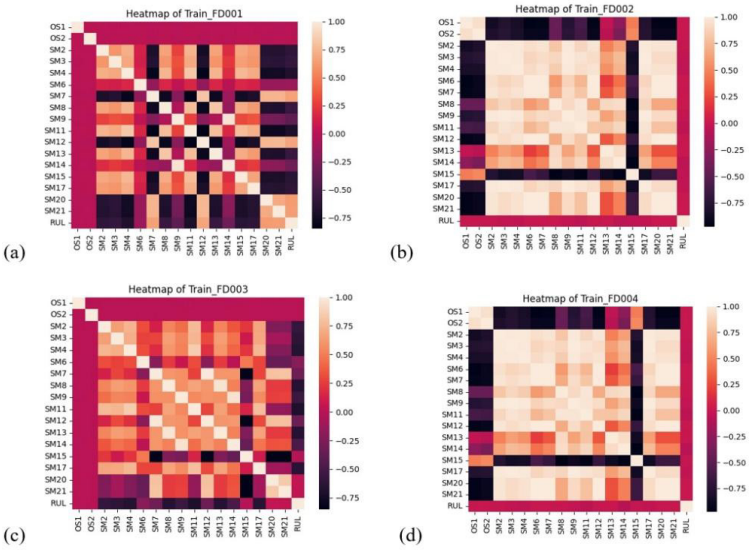

Figure 1. Heatmaps of (a) FD001, (b) FD002, (c), FD003, and (d) FD004 training datasets

Data preparation

The original data contains three operational settings and sensor measurements from column variables 6 to 26, with unclear descriptions. The operational settings are Mach Number (0 to 0.90), altitude (sea level to 40,000 feet), and sea-level temperature (-60°F to 103°F). Upon further research, the labels were identified for the sensor measurements with an explanation of each label provided in Appendix A [9].

From the original C-MAPSS datasets, training sets consist of all data up until each failure, and testing sets have the cut-off of some data before the failure. For example, in the training set for FD001, all rows with unit = 1 are the data from the first failure. The last row of unit 1 has cycles = 192, which means the engine failed after 192 operational cycles. For the testing set, the last row of unit = 1 has cycle = 31, which is not when the failure happened. The separate RUL files contain the number of cycles until failure from the last sample for each test unit. In the case of the test set for FD001 unit 1, it is cycles = 112.

Methodology

Federated learning and FedAvg aggregation

Distributed learning that places a strong emphasis on data privacy and security. It is particularly beneficial in scenarios where data confidentiality is crucial in a highly sensitive environment. This approach was chosen because it helps prevent data leakage and reverse engineering of the data. However, the setup can lead to biased subsets of data at each node, as the training data is not shareable. To address this, a federated learning algorithm is employed. Initially, the server sends instructions to each node to train a local model. These local node models train on their respective data, and after a training round, they only transmit their updated weights to the central server. The central server then aggregates these weights.

There are several aggregation methods available in Federated Learning (FL), each designed to address specific challenges and requirements, such as Federated Average (Fed Avg), FedProx, and personalized FL, etc. FedAvg is one of the most widely used methods, functioning as a generalized version of local-SGD, which performs a weighted average of local model parameters (weights) after a certain number of optimization steps are performed by each model [12]. The weights for the average are determined as seen fit by the implementation. The combined model is then transmitted back to the nodes, which use the updated model parameters as a starting point for another round of training [13]. This process continues for multiple rounds until the global model converges. FedProx is a generalization of FedAvg and is primarily designed to address challenges associated with Independent and Identically Distributed (non-IID) data, which means data distributions vary significantly across devices. This method introduces a proximal term to the local optimization objective, which restricts the local updates to remain close to the global model. This modification improves stability and convergence in heterogeneous environments, making it particularly effective for applications with diverse data sources [14]. Another method is Personalized Federated Learning (FL), an alternative aggregation method that aims to train a model that depends on individual user preferences or behaviors [15]. Unlike FedAvg and FedProx, which aim to train a single global model, personalized FL allows for the creation of customized models for each user or device. This is achieved by incorporating user-specific parameters or fine-tuning the global model locally to better adapt to individual data distributions. Personalized FL is especially useful in scenarios where user data exhibits significant variability, such as in healthcare or recommendation systems.

In our case, we chose to start with a simple method that is FedAvg, where the weight is the number of data samples

https://doi.org/10.61577/jmla.2025.100005

RESEAPRO JOURNALS

available for training at each node. FedAvg is a well-established theoretical foundation approach and simpler to implement than the other approaches, which may require a more complex mechanism for tailoring models to individual users.

Long short-term memory (lstm)

Upon reviewing the current work being done for PM, especially the work done in, we decided that the best technique to model RUL is to use deep learning with LSTM neural network [8]. LSTM is a type of Recurrent Neural Network (RNN) that is suitable for modeling events that happen in sequence. LSTM avoids the common vanishing gradient problem by using gates to store or forget information as needed [16]. This allows LSTM to learn long-term dependencies more effectively while having the ability to forget learned relationships that no longer benefit the goal of minimizing loss. Each LSTM cell is composed of three gates implemented as sigmoid functions. Data x_t comes into the cell, and becomes part of g_t the candidate cell which also has previous memory information. The f_t (forget) gate decides what information should be forgotten; the i_t (input) gate decides what information should be stored, and the o_t (output) gate decides what information should be the output from the cell. Hidden state S_{t-1} and cell state L_{t-1} contain the short and long-term memory states from the previous cells respectively. Figure 2 demonstrates the internal workings of a single LSTM cell.

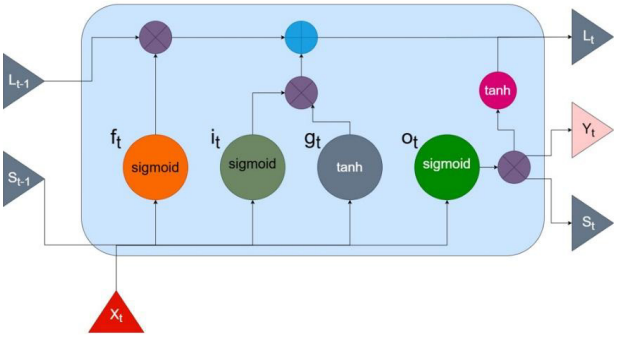

Figure 2. Diagram of LSTM cell internal structures.

Model validation

We used the Keras Tuner python module to do extensive tuning of our neural networks. This module allows nearly limitless customization of model tuning with skilled use of Python. See the documentation of the Keras Tuner Python module for more information on the usage. Table 2 below shows the exact hyper-parameters and values tested.

Table 2. Keras Tuner Configuration

Hyper-parameter Name	Values
Sequence Length	1, 2, 4, 8
Batch Size	1, 8, 32
Layer Dropout	0.1, 0.2, 0.3
Recurrent Dropout	0.1, 0.2
Learning Rate	0.0001, 0.001, 0.002
Gaussian Noise	0.01, 0.1

Metrics and loss

When it comes to metrics for regression problems in machine learning, there are three main loss functions: Mean Squared Error (*MSE*), Mean Absolute Error (*MAE*) and R-squared (R^2). Let N be the number of prediction samples, $\bar{y}$ be the mean of the true values, y_i and $\hat{y}_i$ be the true and predicted values at sample *i*, respectively. These metrics are defined as:

$$MSE = \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N} \quad (1)$$

$$MAE = \frac{\sum_{i=1}^{N}|\hat{y}_i - y_i|}{N} \quad (2)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(\bar{y}_i - y_i)^2} \quad (3)$$

MSE is a very punishing loss because it is non-linear with respect to error size. For example, an error of 10 is 100 times worse than an error of 1 because of the squaring. Additionally, the use of the square changes the units of the error to squared target units which may be difficult to understand. Because of this, Root-MSE is a popular choice: $RMSE=\sqrt{MSE}$. can be good because it is recorded in the same units as the target variable (engine cycles in our case); however, it is linearly punishing and treats an error of $q\beta$ as being q times worse than an error of β which may not be good in certain cases such as jet engine failure. R^2 is generally not chosen as a loss as it is difficult to understand it as a loss function and easy to manipulate by adding more variables to our input and "curating" the validation data to have low variance. However, given two models with the same input variables evaluated on the same data, R^2 can be a good "tie-breaker" for similar *RMSE* or *MAE*. For the reasons laid out above, we selected to use RMSE for the training and validation loss of our models, but we reported *MAE* as our second metric.

Model architecture

Derived from the work done in, we used a neural network architecture in Figure 3 that leverages the recurrent capabilities and robustness of LSTM cells with the approximation abilities of fully connected layers [8]. The dropout layers and recurrent dropout in the LSTM are used to prevent over-fitting and improve generalization on unseen data. The Gaussian noise layers are also generalization-aiding layers, as well as the differential privacy facet of Federated Learning.

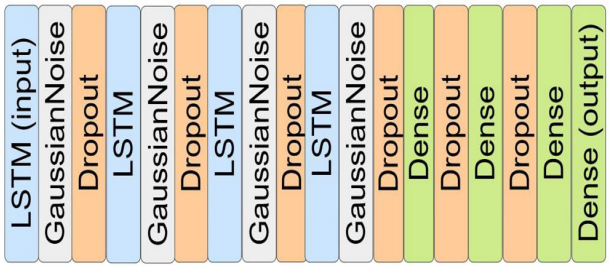

Figure 3. Neural network architecture diagram

Preprocessing

Preprocessing data effectively is crucial for ensuring optimal performance in machine learning models, and it involves three key techniques: filtering, pruning units, and scaling.

https://doi.org/10.61577/jmla.2025.100005

RESEAPRO JOURNALS

Many of the signals that make up the input of this dataset have noise included. The noise was purposefully added when the data was being generated [9]. Noise can be bad for machine learning models since the models may over-fit the noise of the training data, which has no real influence on the underlying phenomena. Filtering is a way to reduce the noise in the signals and improve the model generalization. We chose to use a median filter, which is a sliding window filter that suppresses noise and outliers, as illustrated in Figure 4 for Engine 11 of FD002 [17,18].

To select the kernel size for filtering each signal, we divided the signals based on each unit within each node. Then we tried kernel sizes k=3,5,7,9,11,…, M_k where $M_k = \frac{s_i}{[10]}$, s_i being the number of samples for unit i. The best kernel for each unit was selected as the one that maximizes the linear correlation between the target (RUL) and the signal being analyzed. We then took the weighted geometric average of all the kernels and set it as the final kernel size to use. The weight varied as the correlation and the number of samples for the unit. The best was selected depending on the final model performance.

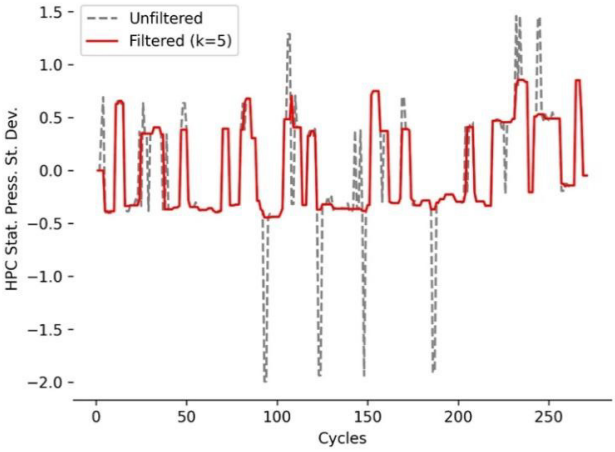

Figure 4. Example of Median Filter on HPC Static Pressure Signal for FD002 Engine 11 Time Series.

The second technique used is a simple "pruning" technique, which randomly removes some samples from the ends of the sequences of some of the time series in the training set. This method is similar in execution to neural network pruning, though the goal is to make the data more representative as opposed to reducing complexity [19,20]. This was done to mimic the test set time series, which did not run until failure. The general idea was to prevent the model from over-fitting to a time series that starts at full health and ends at critical failure. Listing 1 is the code that was used for pruning.

Scaling inputs are important when dealing with data in vastly different scales, such as data taken from different sensors from different jet engines in different operating conditions [21]. The differing scales and variances can cause the neural network to unfairly bias certain dimensions in the data due to undesirable reasons. We elected to use the popular z-score normalization technique, which takes all dimensions in the data and normalizes them to have approximately zero mean and standard deviation of 1. This creates an even playing field when it comes to assigning neural network parameters during optimization. To use this technique, we simply turn each observation in each sample into a z-score. Let X be the set of all observations for a certain feature in one of our agents, we can find the z-score for the i^{th} observation x^i using:

$z_i = \frac{xi - \bar{X}}{\sigma_x}$ from noise, outlier, and scale-related biases.

Listing 1: Pruning Units

```
import numpy as np
import pandas
def prune_units(df,
                key="unit",
                prune_chance=0.3,
                p=0.4, pplus=0.1):
    for _, data in df.groupby(key):
        chance = np.random.rand()
        Rows = data.shape[0]
        if chance <= pplus:
            prune rows = rows*0.75
            df.drop(
            data.tail(int(
                      prune_rows
                )).index,
                inplace=True)
        elif chance <= prune_chance:
            prune_rows = rows*p
            df.drop(
            data.tail(int(
                      prune_rows
                )).index,
                inplace=True)
```

Listing 1. Pruning units

Feature engineering

Given that our dataset is made up of multiple time series that deal with the degradation of parts, we decided to use some feature engineering techniques to add dimensions and new predictors to our data. The first technique is a simple accumulation technique where we take the cumulative sum of certain signals throughout the degradation process. The second technique is explained as follows.

As proposed by Mu et al., taking the derivative of the input space can add helpful dimensions to our dataset [22]. We took a similar approach by taking a simple, single-dimension, single-time-step signed change calculation to some of our features and added these as new features. The single time step is because we do not know how long the "cycle" time dimension is. Given that most engines in the dataset fail within 200 cycles, it is safe to say each cycle is a relatively large time step, such as hours or even an entire flight. Thus, having too large of a window may not capture meaningful change. The following snippet of code in Listing 2 demonstrates how the rate of change is calculated for one dimension and time-step = dt:

https://doi.org/10.61577/jmla.2025.100005

Listing 2: Rate of Change

```
import numpy as np
import pandas as pd

def derive(data, feature, dt):
    x = data[feature].to_numpy()
    dx = data[feature].to_numpy()
    # first dt-1 samples assumed
    # to have no change in x
    dx[:dt] = 0
    for i in range(dt,data.shape[0]):
        dx[i] = (x[i] - x[i-dt])/dt
    return dx
```

Listing 2. Rate of Change

Results and Discussion

Model hyperparameters

The hyperparameter tuning results in Table 3 outline the optimal configuration for the Long Short-Term Memory (LSTM) network used in predicting the Remaining Useful Life (RUL) of jet engines within a federated learning framework. The model employs 4 LSTM layers and 4 dense layers, each with 64 units per layer, which balances model complexity and computational efficiency. A batch size 32 was selected, enabling efficient training while maintaining sufficient gradient updates. To prevent overfitting, a dropout rate of 0.1 and a recurrent dropout rate of 0.2 were applied, ensuring robust generalization to unseen data. The learning rate was set to 0.001, a standard choice for achieving stable convergence during training. Additionally, Gaussian noise with a variance of 0.01 was introduced to improve the model's generalization by simulating slight variations in the input data. These hyperparameters were fine-tuned through extensive experimentation using KerasTuner, ensuring optimal performance for the predictive maintenance task while preserving data privacy in a federated learning setup.

Table 3. Hyperparameter selection

Hyper-parameter Name	Best Values
LSTM Layers	4
Dense Layers	4
Units Per Layer	64
Batch Size	32
Layer Dropout	0.1
Recurrent Dropout	0.2
Learning Rate	0.001
Gaussian Noise	0.01

Model performance

Below is Table 4 detailing the validation and testing set performance for our agents and our aggregated model. We also have graphical demonstrations in Figure 5 of how our model predicts the degradation of the different faults over time, considering the different conditions simulated within the different agents.

Table 4. Validation and Test Performance

Agent	Validation RMSE	Test RMSE
FD001	13.7014	13.811
FD002	14.6618	20.587
FD003	14.3186	14.201
FD004	18.1955	22.998
Aggregated	15.2193	17.899

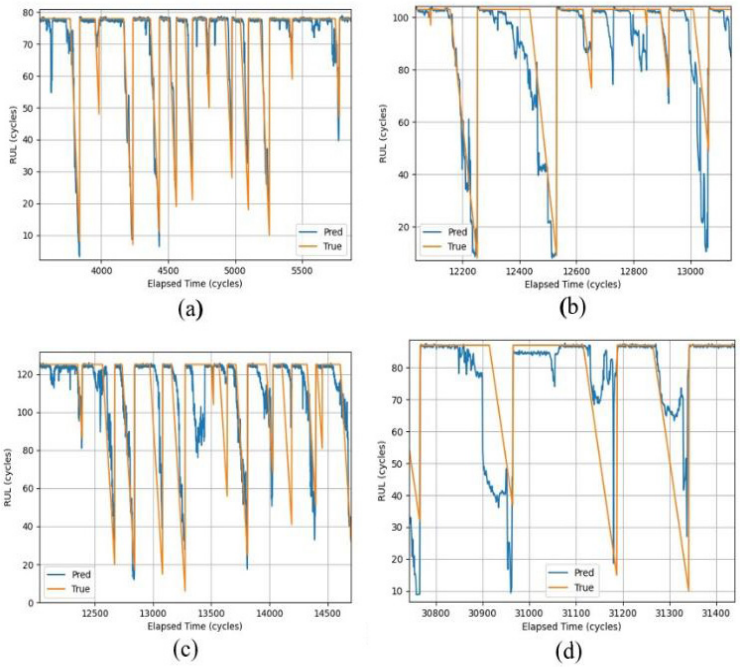

Figure 5. Testing set predictions for (a)FD001, (b)FD002, (c)FD003, and (d)FD004.

Statistical Analysis

We averaged the results across the different agents over several studies aggregated by Asif et al. and compared the results with our federated averaged model results [1]. We used statistical tests to compare these models. First, we ran two-tailed, equal means tests for all of them without selecting an α value beforehand. All the p-values were extremely low. Thus, all of the equal means hypotheses were rejected, suggesting that our models either outperformed or underperformed against each study.

We then performed greater-than-or-equal-to and less-than-or-equal-to one-tailed tests against each of the studies and recorded the results. Table 5 below summarizes the studies for which our model was statistically better on average. The study number refers to the order in which the study appears in Table 9 of the paper by Asif et al. [1].

https://doi.org/10.61577/jmla.2025.100005

Table 5. Statistical Model Comparison

Study	Null Hypothesis (H_0)	t statistics	Reject H_0?
13	$\mu \leq 18.44$	-14.35	No
3	$\mu \leq 18.86$	-23.70	No
5	$\mu \leq 19.24$	-32.37	No
10	$\mu \leq 19.29$	-33.39	No
2	$\mu \leq 21.24$	-77.50	No

Note: all p-values 0.99999999 or higher.

We also generated 95% confidence intervals for the true mean error of our agent and aggregated models in Table 6.

Table 6. Confidence Intervals for Mean Model Errors

Agent Name	Lower Bound	Upper Bound
FD001	13.10	13.44
FD002	20.55	21.16
FD003	13.82	13.98
FD004	22.90	23.48
Aggregated	17.71	17.90

Discussion

The experiment results validated our hypothesis, as, on average, the FL approach either maintained or improved the model performance. While the test error across the individual agents showed significant variance, the aggregated model demonstrated the ability to generalize and balance the error overall. The variance in individual agent performance can be explained by the operating conditions detailed in Table 1. FD002 and FD004 are simulated with variance in six operating conditions (altitude, pressure, air speed, etc.), while FD001 and FD003 are simulated at sea level with no variance. Thus, FD002 and FD004 performed 1.25-1.6 times worse than the other agents. The statistical analysis provided further insights into the effectiveness of the FL model. By conducting two-tailed equal means tests and greater-than-or-equal-to and less-than-or-equal-to one-tailed tests, we were able to compare our model's performance against other studies. The consistently low p-values and the rejection of null hypotheses in several cases suggest that our FL-based model either outperforms or is comparable to existing methods in predictive maintenance. This statistical rigor confirms that the federated approach not only preserves data privacy but also maintains, if not improves, model accuracy.

Conclusions

This study explores the application of FL in PM tasks, focusing on engine fault prediction using deep learning with LSTM cells. The primary motivation behind this work was to address the challenges of maintaining data privacy while allowing collaboration between entities with data silos. Our main hypothesis was that implementing the FL framework would either maintain or improve the model performance on average when compared to the centralized learning approach. The implementation of FL offers several key advantages in the context of predictive maintenance. Firstly, it preserves data privacy by ensuring that sensitive data remains localized on the devices where it is generated. This is particularly important in industries like aerospace, where operational data can include proprietary and sensitive information. By minimizing the need for data centralization, FL reduces the risk of data breaches while allowing collaboration between entities that would otherwise not be able to collaborate. However, the study also highlights some challenges associated with the implementation of FL in predictive maintenance. Managing the FL process across multiple devices introduces complexity, particularly in coordinating updates and maintaining consistent model performance across diverse environments. Additionally, the federated approach can lead to biased subsets of data at each node, which may affect the generalization of the global model. Despite these challenges, the ability to collaborate across different entities without sharing raw data opens new possibilities for improving the maintenance models industry-wide. This collaborative approach can accelerate advancements in predictive maintenance technologies, benefiting not just individual companies but the entire industry.

Declarations

This project has no external funding, and all authors consent to its publication. The data was acquired from NASA PCoE under a Public Domain License, and the code is available upon request.

References

1. McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. InArtificial intelligence and statistics; 2017. 1273-1282p. https://doi.org/10.48550/arXiv.1602.05629
2. Diamoutene A, Kamsu-Foguem B, Noureddine F, Barro D. Prediction of US General Aviation fatalities from extreme value approach. Transp Res A: Policy Pract. 2018;109:65-75. https://doi.org/10.1016/j.tra.2018.01.022
3. Yin X, Zhu Y, Hu J. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. ACM Comput Surv. 2021;54(6):1-36. https://doi.org/10.1145/3460427
4. Tene O, Polonetsky J. Privacy in the age of big data: a time for big decisions. Stan. L. Rev. Online. 2011;64:63.
5. Stanton I, Munir K, Ikram A, El-Bakry M. Predictive maintenance analytics and implementation for aircraft: Challenges and opportunities. Syst Eng. 2023;26(2):216-237. https://doi.org/10.1002/sys.21651
6. Meissner R, Rahn A, Wicke K. Developing prescriptive maintenance strategies in the aviation industry based on a discrete-event simulation framework for post-prognostics decision making. Reliab Eng Syst Saf. 2021;214:107812. https://doi.org/10.1016/j.ress.2021.107812
7. Jiang Y, Tran TH, Williams L. Machine learning and mixed reality for smart aviation: Applications and challenges. J Air Transp Manage. 2023;111:102437. https://doi.org/10.1016/j.jairtraman.2023.102437
8. Asif O, Haider SA, Naqvi SR, Zaki JF, Kwak KS, Islam SR. 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
9. Saxena A, Goebel K, Simon D, Eklund N. Damage propagation modeling for aircraft engine run-to-failure simulation. In2008 international conference on prognostics and health management; 2008. 1-9p.
10. Botre M, Brentner KS, Horn JF, Wachspress D. Validation of helicopter noise prediction system with flight data. InVertical Flight Society 75th Annual Forum & Technology Display; 2019. https://doi.org/10.4050/F-0075-2019-14447

https://doi.org/10.61577/jmla.2025.100005

11. Ebrahimi A, Jafari S, Nikolaidis T. Heat load development and heat map sensitivity analysis for civil aero-engines. Int J Turbomach Propuls Power. 2024;9(3):25. https://doi.org/10.3390/ijtpp9030025
12. Wang J, Charles Z, Xu Z, Joshi G, McMahan HB, Al-Shedivat M, et al. A field guide to federated optimization. arXiv preprint arXiv:2107.06917. 2021. https://doi.org/10.48550/arXiv.2107.06917
13. Qi P, Chiaro D, Guzzo A, Ianni M, Fortino G, Piccialli F. Model aggregation techniques in federated learning: A comprehensive survey. Future Gener Comput Syst. 2024;150:272-293. https://doi.org/10.1016/j.future.2023.09.008
14. Mu X, Shen Y, Cheng K, Geng X, Fu J, Zhang T, et al. Fedproc: Prototypical contrastive federated learning on non-iid data. Future Gener Comput Syst. 2023;143:93-104. https://doi.org/10.1016/j.future.2023.01.019
15. T Dinh C, Tran N, Nguyen J. Personalized federated learning with moreau envelopes. Adv Neural Inf Process Syst. 2020;33:21394-1405.
16. Noh SH. Analysis of gradient vanishing of RNNs and performance comparison. Information. 2021;12(11):442. https://doi.org/10.3390/info12110442
17. Kaur H, Virmani J, Thakur S. A genetic algorithm-based metaheuristic approach to customize a computer-aided classification system for enhanced screen film mammograms. InU-Healthcare Monitoring Systems; 2019. 217-259p. https://doi.org/10.1016/B978-0-12-815370-3.00010-4
18. Kumar A, Sodhi SS. Comparative analysis of gaussian filter, median filters and denoise autoenocoder. In2020 7th international conference on computing for sustainable global development (INDIACom); 2020. 45-51p. https://doi.org/10.23919/INDIACom49435.2020.9083712
19. Li L, Zhu J, Sun MT. Deep learning based method for pruning deep neural networks. In2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW); 2019. 312-317p. https://doi.org/10.1109/ICMEW.2019.00-68
20. Pasandi MM, Hajabdollahi M, Karimi N, Samavi S. Modeling of pruning techniques for deep neural networks simplification. arXiv preprint arXiv:2001.04062. 2020. https://doi.org/10.48550/arXiv.2001.04062
21. Sharma V. A study on data scaling methods for machine learning. Int J Glob Acad Sci Res. 2022;1(1):31-42. https://doi.org/10.55938/ijgasr.v1i1.4
22. Mu X, Pavel AB, Kon M. Differentiation and integration of machine learning feature vectors. In2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA);2016. 611-616p.

https://doi.org/10.61577/jmla.2025.100005