IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Journal Européen des Systèmes Automatisés
Vol. 59, No. 2, February, 2026, pp. 315-324
Journal homepage: http://iieta.org/journals/jesa

Split Federated Learning for Efficient and Privacy-Preserving Predictive Maintenance in Resource-Constrained Industrial IoT Systems

Haneen H. Adel[1*], Osamah F. Abdulateef[1], Ali H. Hamad[2]

[1] Automated Manufacturing Engineering Department, AL Khwarizmi College of Engineering, University of Baghdad, Baghdad 10071, Iraq
[2] Biomedical Application, College of Artificial Intelligence, University of Baghdad, Baghdad 10071, Iraq

Corresponding Author Email: Hanin.Adel2404m@kecbu.uobaghdad.edu.iq

https://doi.org/10.18280/jesa.590203

Received: 12 December 2025
Revised: 9 February 2026
Accepted: 24 February 2026
Available online: 28 February 2026

Keywords:
Split Learning, Federated Learning, internet of things, predictive maintenance, Split Federated Learning

ABSTRACT

Two popular distributed model training techniques allow clients to test and train models without sharing raw data, namely Split Learning (SL) and Federated Learning (FL). Because the model is architecturally separated between the server and clients, SL offers improved model privacy and is therefore better suited for environments with restricted resources. Yet, due to its relay-based, sequential training method, SL frequently performs worse than FL. For the purpose of overcoming such restrictions, this study presents Split Federated Learning (SFL), which is a new hybrid framework that minimizes the inherent shortcomings related to SL as well as FL, and combines their benefits. The initial layers regarding the model are processed locally by each client device, like a controller or sensor node, and the remaining layers are calculated on a main server in the suggested structure. Differential privacy is incorporated into the architecture of SFL to improve the robustness of the model and enhance data protection. The SFL framework allows several edge devices to work together for jointly training models for machine condition monitoring in predictive maintenance (PdM) in industrial systems, all while protecting the sensitive operational data. Also, the experimental tests show that SFL dramatically reduces calculation time per global epoch with regard to multiple clients, and achieves similar communication efficiency as SL. Furthermore, SFL shows better stability, also maintaining high communication efficiency in comparison with FL when the number of clients rises. Because of such features, SFL is a dependable as well as expandable option for putting PdM into Industrial Internet of Things (IoT) settings.

1. INTRODUCTION

A method called predictive maintenance (PdM) uses real-time data for maximizing equipment utilization as well as reducing needless repairs. PdM allows for the scheduling of interventions before the predicted occurrence of a failure through continuously assessing and monitoring the health of the system online [1, 2]. When combined, Industrial Internet of Things (IIoT) and PdM constitute a paradigm shift in equipment management, especially when time series data from the devices of IIoT devices is analyzed using deep learning (DL) [3]. A vast array of application fields is being profoundly impacted by IoT, a scalable network system of dispersed connectivity and devices. Intelligent industrial operations, enhancing operational efficiency as well as production processes, are made possible through the growing volume and velocity of data produced through IIoT devices in conjunction with the enhanced analytics capabilities, especially in Industry 4.0 [4]. There are four primary categories related to industrial maintenance: Through predicting potential issues before they arise, PdM extends the longevity of equipment, as well as reduces unscheduled machine downtime. In addition, Proactive Maintenance (PRM) attempts to prevent failures through tackling their underlying causes, whereas Reactive Maintenance (RM) concentrates on repairing the equipment when it breaks down, implying that downtime could be substantial due to the unpredictability of failure duration. For this reason, PM is performed regularly, in spite of the actual state of the machine. As opposed to reactive, which is less expensive, yet incurs greater costs if equipment malfunctions, causing unplanned downtime. While preventative measures could lead to needless maintenance, predictive solutions allow for planning and reducing the unexpected downtime. PdM just reduces unplanned downtime due to the fact that it involves condition monitoring, whereas the preventative measures could lead to unnecessary maintenance [5]. Also, PdM is just performed as necessary, since it involves condition monitoring. This must lead to a reduction in the requirement for replacement parts, hence saving money. It also aids in avoiding over-maintenance, which usually results in issues. PM could involve reevaluating systems for avoiding breakdown, whereas PdM mostly concentrates on scheduling maintenance for equipment problems. There are more advantages to utilizing modern technology, like machine learning (ML), IoT sensors, and data analysis. Forecasts grew more accurate as a result. Early problem-solving could extend

the longevity of equipment. Real-time data analysis as well as storage are essential for PdM, taking into consideration the various aspects and effects related to the signals collected. Time series analysis, artificial intelligence (AI), multiple failure modes, and data analytics are only a few of the methods used in PdM [6].

For the purpose of assessing the high-dimensional sensor data, contemporary PdM techniques mostly depend on the models of DL, like recurrent neural networks (RNNs) as well as convolutional neural networks (CNNs) [7]. Despite their great accuracy, such techniques frequently call for the aggregation of raw data on a central server, which raises questions around data bandwidth and security constraints in extensive IIoT implementations [8]. Distributed learning models, such as SL and FL, have surfaced as privacy-preserving substitutes for addressing such problems. Although FL makes it possible to train models collaboratively over edge devices with no sharing of raw data, it has problems with heterogeneity as well as significant communication overhead [9]. SL improves privacy and reduces bandwidth consumption through dividing deep neural networks across servers and clients and sending just intermediate activations (smashed data) [10]. The main problem with SL, in spite of its benefits, is that its relay-based training causes clients' resources to stay idle since just one client interacts with the server at a time, which significantly raises training overhead when there are numerous clients. SFL, which is a hybrid paradigm combining the benefits of SL and FL, has lately surfaced as a solution to such constraints, providing improved scalability, efficiency, and privacy protection. SFL emphasizes model robustness as well as data privacy, taking into account the benefits regarding SL and FL [11]. It has been observed that SFL provides a superior solution with better model privacy in comparison with FL, faster than SL, and comparable communication efficiency as well as model accuracy to SL. All things considered, SFL is advantageous in the resource-constrained settings, in which complete model deployment and training are not practical, and quick model training time is needed for updating the global model regularly using a dataset that is continuously changing over time (data stream) [12].

This research's primary contribution is filling the current research gap on SFL and SL in IIoT systems. The presented study demonstrates how SFL and SL can enhance communication effectiveness, data privacy, and security in dispersed industrial systems, whereas the majority of prior research has mostly concentrated on traditional FL designs. Additionally, the research offers a workable framework incorporating such learning paradigms into PdM systems, depending on IoT, allowing for safe cooperation with little exposure to data as well as improved model performance. Furthermore, the study uses optimized datasets to provide a thorough comparison analysis regarding SL, FL, SFL, and standalone learning. SFL can be considered the most significant method for attaining effective and secure learning in the IoT-enabled industrial settings, according to the review, which highlights the trade-offs between security and performance among various paradigms. We believe that research on SFL and SL, specifically on their effectiveness as well as suitability for industrial PdM systems, is currently lacking. The key contributions of this work are now summarized as follows:

- Designing a Split Federated Learning (SFL) framework for PdM in resource-constrained IIoT environments.
- Developing an experimental IoT testbed using Raspberry Pi devices and multiple sensors for real-time machine condition monitoring.
- Providing a comparative evaluation between SL, FL, and SFL frameworks using the collected motor dataset.

2. THEORETICAL AND BACKGROUND

One of the major uses related to AI in smart manufacturing, as well as IIoT, is PdM, which is a fundamental component of Industry 4.0. For minimizing unscheduled downtime and maintenance expenses, PdM focuses on predicting machine failures using multi-sensor data, including temperature, vibration, and current signals.

2.1 Split Learning

About distributed ML, SL is considered a cooperative technique enabling many clients to train a common model without sharing raw data [13]. The technique divides the neural network (NN) at a certain layer called the cut layer. Just the intermediate output, known as smashed data, is sent to the server by each one of the clients after processing the first few layers of the model locally [14]. After that, the server computes the backpropagates, loss, and continues the computation on the remaining layers. In addition, the client then updates its local model segment using the gradients of the cut layer that the server has returned to it [15]. The two sides contribute to the model's overall training using such a back-and-forth process, which protects as well as preserves sensitive data [16]. A total of three client devices (client A, client B, and client C) are used in Figure 1 to show how SL operates. Raw data from clients is never seen by the server. Just the smashed data is sent to the server by each one of the clients, which trains its part regarding the model up to a specific layer known as the cut layer. For clients to update their local models, the server completes the forward pass, computes the loss, and returns gradients. Until the model is completely trained, the exchange is carried out over all client data in several training rounds.

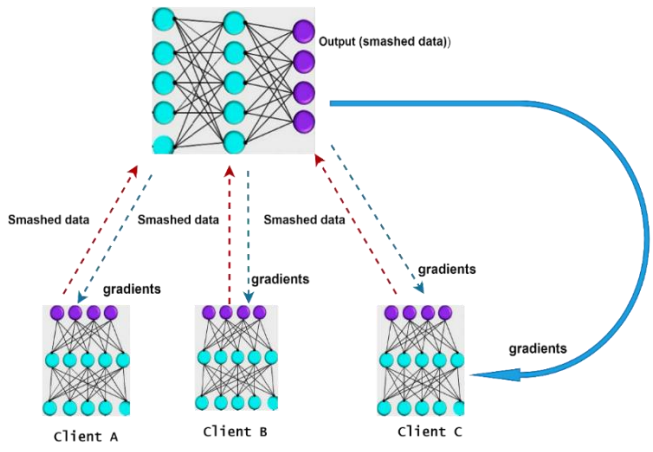


Figure 1. Split Learning (SL) representation with three client devices

Two formal functions could be distinguished in the SL model:

$$f(x) = f_s(f_c(x)) \quad (1)$$

where:

- $f_c(\cdot)$ denotes the layers executed on the client side,
- $f_s(\cdot)$ denotes the layers executed on the server side,
- x denotes input data.

Throughout the forward pass, the client computes:

$$a = f_c(x) \tag{2}$$

Here, a (the smashed data) is sent to the server, which then produces the prediction:

$$\hat{y} = f_s(a) \tag{3}$$

The gradients are backpropagated, and the loss is calculated through the server. The gradient of the cut layer, $\nabla_a L$, is sent back to the client, which updates its local parameters:

$$w_c = w_c - \eta \nabla_{w_c} L(y, \hat{y}) \tag{4}$$

Meanwhile, the server updates its parameters as:

$$w_s = w_s - \eta \nabla_{w_s} L(y, \hat{y}) \tag{5}$$

2.2 Federated Learning

A distributed ML technique called FL allows several clients to work together to create a common model with no sharing of the raw data. By keeping the data local as well as just sharing model updates across the network, it lowers the communication overhead and also provides a number of important benefits, including data privacy, efficiency, security, and scalability. Federated Averaging (FedAvg), which is considered a model aggregation process used in ML [17], is performed on the server, in which a global model is produced through averaging local model updates. FL trains a shared global model with three clients, client A, client B, and client C, as illustrated in Figure 2 [18]. The model parameters are aggregated by the central server, and each one of the devices updates its own model solely with its own dataset. Raw data never leaves the clients; just the model updates (gradients or weights) are transferred, permitting collaborative training, and maintaining the privacy of data. Yet, when devices have varying capabilities, as is typical in IoT systems, training the entire model on the devices with restricted resources could be computationally demanding as well as slow [19, 20]. Notwithstanding such difficulties, FL offers a workable method for developing PdM models at several locations without disclosing private information [21]. FL's formal objective is to minimize the global objective function, which is specified as follows:

$$f(w) = \frac{1}{n} \sum_{j=1}^{n} f_j(w) \tag{6}$$

where, each $f_{j(w)}$ denotes the empirical risk computed on the client $j's$ dataset. For a given j, this local object could be indicated as:

$$f_j(w) = \frac{1}{|D_j|} \sum_{(x,y) \in D_j} \ell\,(x, y, w) \tag{7}$$

Here D_j represents the local dataset of client j, $\ell(x, y, w)$ is the chosen loss function. The most widely adopted aggregation algorithm in FL is Federated Averaging (FedAvg), where the global model is updated as follows:

$$w^{(t+1)} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^{(t)} \tag{8}$$

with $w_k^{(t)}$ denoting the weights regarding the local model trained through the client k, n_k is the number of local samples at client k, $n = \sum_{k=1}^{K} n_k$ is the total number of samples over all clients.

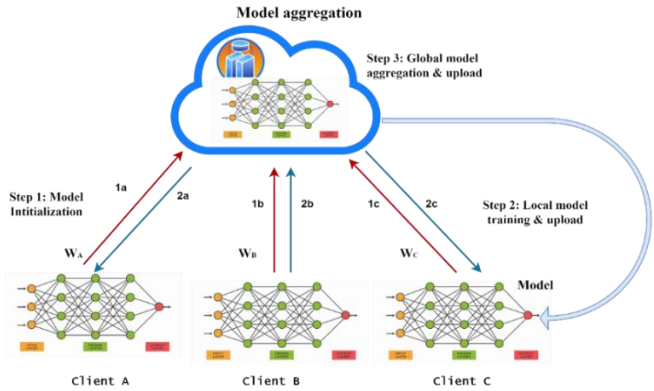


Figure 2. Federated Learning (FL) representation with three devices

2.3 Split Federated Learning

Lately, a significant paradigm combining the advantages of SL and FL has surfaced: SFL. SFL lessens the computational load on edge devices and maintains the privacy of data through dividing processing between servers and clients [22]. This method is especially crucial for IoT-enabled settings, in which the privacy of data is a top priority and devices have limited resources. Every client sends their smashed data to the main server, and concurrently does forward propagation on their client-side model, including their noise layer. After that, the main server uses its server-side model for processing forward as well as back-propagation independently, in (relatively) parallel, using smashed data from each one of the clients. After that, it forwards the smashed data's gradients to the appropriate clients for back-propagation. The server then uses FedAvg, or weighted average regarding gradients calculated throughout back-propagation on each one of the client's smashed data, for updating its model. Each one of the clients computes its gradients at the client's end after back-propagating on its client-side local model, following obtaining gradients of its smashed data. The gradients are sent to the fed server. Furthermore, all participating clients receive FedAvg regarding client-side local updates, which is performed by the Fed server [23]. Table 1 compares FL, SL, and SFL, whereas Figure 3 depicts the architecture of SFL. FL's global purpose could be stated formally as follows:

$$f(w) = \frac{1}{n} \sum_{j=1}^{n} f_j(w) \tag{9}$$

In SFL, each client's local model is split into two parts.

$$f(x) = f_s(f_c(x)) \quad (10)$$

During training, the client computes.

Finally, instead of keeping each client's model separate, SFL combines them using federated averaging to update a shared global client-side model.

$$w^{(t+1)} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^{(t)} \quad (11)$$

where, $n_j$is the local dataset size of client j and $n_j = n \sum_{j=2.}^{k}$

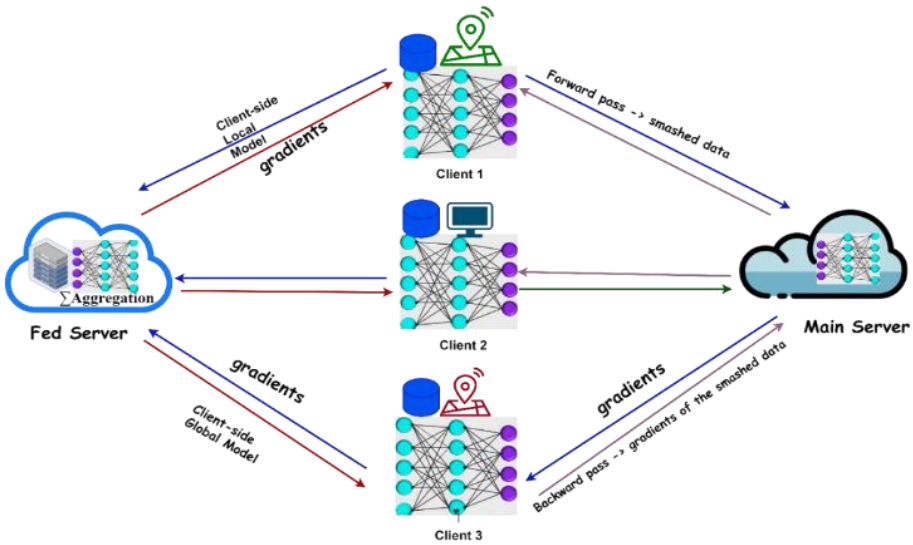


Figure 3. Overview of the Split Federated Learning (SFL) system model

Table 1. An abstract comparison of Federated Learning (FL), Split Learning (SL), and Split Federated Learning (SFL)

Framework	Aggregation Method	Security / Privacy	Computation on Clients	Communication Overhead	Model Size	Notes
SL	Centralized on the server	High – only intermediate activations are shared	Low – only initial layers trained locally	High–frequency exchange of activations and gradients	Small – only client-side portion of the model	Preserves privacy, but slower due to sequential communication
FL	Federated averaging	Medium-High – only model updates are transmitted	High – full model trained on each client	Medium – transmission of model weights	The complete model resides on each client	Efficient for capable devices, may be demanding for low-resource clients
SFL	Federated averaging + split model	High – raw data never leaves the client	Medium – partial model trained locally, aggregated server-side	Low – combines smashed data exchange with federated aggregation	Medium – partial client model + server-side portion	Balances privacy, computation, and communication efficiency, offering the best overall performance

3. DEEP NEURAL NETWORKS

Table 2. Hyperparameters of the proposed lightweight model

Parameter Name	Value
Number of neurons per layer	50
Batch size	64
Epochs	25
Loss function	Categorical cross-entropy
Activation function	ReLU, SoftMax (in output layer)
Optimizer	Adam
Number of classes (output layer)	5
Number of layers	5
Cut layer	32

The hyperparameters for the proposed DNN model are depicted in Table 2. The model consists of five layers. The input layer is equal to the number of input features, while the output layer is equal to the number of classes, which is five in this case [24].

4. DATASET IMPLEMENTATION

For collecting sensor data, the presented study makes use of a variety of sensors, such as contactless temperature sensors, 3-axis accelerometers, current sensors, and ambient temperature and humidity sensors. The vibration sensor successfully recorded a defect status per one-second sample time by producing three unique features corresponding to the Y, X, and Z axes. This resulted in multiple columns inside the same dataset. Offline as well as online data collecting methods were used for recording certain fault categories, including

vibration, high load, high current, stop rotation, and normal behavior. For providing the required dataset for every form of fault, such criteria have been carefully created in the standard system. An imbalanced mass has been used for replicating vibrations in all three axes of the motor. Capacitors of various sizes have been used to simulate changes in motor current, and a mechanical brake system was used for simulating heavy loads as well as stop situations. The distribution and quantity of datasets for each one of the fault categories are shown in Table 3. The four categories of faults were categorized in the presented study using a deep neural network model, with the fifth class representing normal behavior. Before applying the DNN-based SFL algorithm, a dataset must be preprocessed, such as normalizing and labeling the different classes as follows:

Table 3. Class sizes and counts

Failure Type	File Size (KB)	Data Records
Normal (No failure)	813 KB	20059
High-current	202 KB	4591
Vibration	301 KB	6562
Stop rotate	651 KB	16566
High-load	140 KB	2911

i. Normalization: Dataset normalization is a curried pre-processing step in ML. It involves transferring the feature in the dataset to common scales, typically between send 1 or -1.

ii. Preprocessing for dataset labeling includes converting categorical features into a numerical representation suitable for ML models. This transformation utilizes a one-hot encoder to generate a binary column (values of either 0 or 1) for every distinct category within the original data.

iii. Dataset splitting: The Dataset has been split into 70% training and 30% testing, where the training set is used to train the model, which is normally the largest portion of data. Testing is used to assess the fine-tuned performance of the trained model on unseen data. This provides an estimate of how well the model generalizes. Figure 4 shows the collected dataset distribution for five types of classes, namely normal, high-current, high-load, stop, and vibration. As shown in this figure, the class is not uniformly distributed.

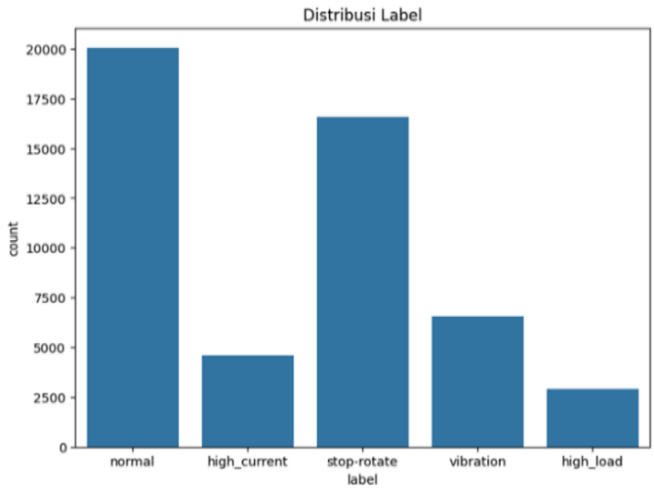


Figure 4. Dataset distribution

iv. Class balance: To resolve class imbalance in datasets, in this work, the Synthetic Minority Over-sampling Technique (SMOTE) is used. This gives a DNN model a more balanced perspective on the issue and increases the likelihood that it will learn to categorize all classes efficiently by boosting the representation of minority classes in training data. This contributes to the confirmation that the SMOTE operation was effective in achieving more equitable class allocations. Figure 5 shows the five classes after balance.

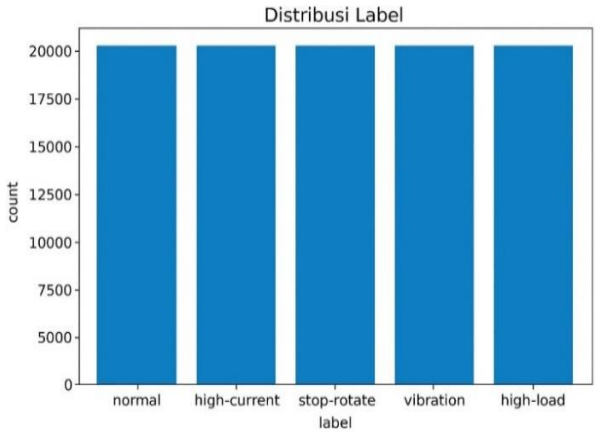


Figure 5. The distribution of the five datasets

5. THE PROPOSED SYSTEM DESIGN

With regard to PdM applications in industrial settings, the suggested framework offers an SFL architecture. A total of three motors, each one of them with four sensors a three-axis vibration sensor, a contactless infrared sensor, two temperature sensors, a current sensor, along with an ambient sensor make up the suggested typical testbed. local DNN model, or three DNN models or clients, is present on a Raspberry Pi. A total of three main computational components are integrated into the system: the Main Server, the Fed Server, also client-side nodes. Layers of neurons make up a deep neural learning DNN model; fifty neurons at the input layer representing each one of the dataset columns, and five neurons at the output layer representing each one of the system classes. The output (cloud) is (5), the cut layer is (32), the round robin technique is (5), and several features are present when the model is being trained. The three operating modes related to the suggested system are major models training, as well as predictive mode, local, and global. A realistic dataset from various motor types-which have been purposefully created-was collected before model training. Every client sends their smashed data to the main server while concurrently doing forward propagation on their client-side model, such as their noise layer. After that, the main server uses its server-side model for processing the forward as well as back-propagation independently, in (relatively) parallel, using smashed data from each one of the clients. After that, it forwards the smashed data's gradients to the appropriate clients for back-propagation. The server then uses FedAvg, or a weighted average of gradients calculated throughout back-propagation on each one of the client's smashed data, for updating its model. Each one of the clients computes its gradients at the client's end after back-propagating on its client-side local model, following obtaining the gradients regarding its smashed data. The gradients are sent to the fed server. All the participating clients receive FedAvg regarding client-side local updates, which is performed by the Fed server. Also, the DNN model

is split into three main parts in the suggested SFL framework for effectively balancing the privacy protection, communication optimization, and computational distribution. Each of such segments-the Fed Server, the Main Server, and Client-Side arrangement-performs a distinct function within the training process as a whole. The suggested system design is depicted in Figure 6.

Client-Side: Each client device, implemented using a Raspberry Pi 4, operates as an intelligent sensing and preprocessing unit attached to an industrial motor. In which Raspberry Pi-4's I2C communication protocol is used to interface MLX90614, SHT21, and ADXL345 sensors. ADC, converting the current value to digital form, must be connected to ACS712. I2C, a communication technology, where each one of the sensors has its own distinct address for preventing packet collisions, was used to link all of these sensors with three Raspberry Pi 4s for each one of the motors. The MQTT protocol is used by each Raspberry Pi for connecting to the cloud server. After collecting data from several sensors, like vibration, temperature, and current sensors, it carries out preliminary preprocessing operations, like normalization, feature extraction, and filtering. For converting raw sensor readings into the intermediate feature representations (activations), the client runs the first section regarding the split DNN. After that, such activations are safely sent to the Main Server for additional processing. This approach minimizes bandwidth usage and ensures that sensitive raw data remains localized at the machine level.

Main Server: The main server acts as the computational partner for all client nodes, hosting the second segment of the DNN model. It performs advanced representation learning and manages the backpropagation process. The data flow between the components follows the sequence:

Clients → (activations) → Main Server → (gradients) → Clients.

Client gradients → Fed Server → Aggregated client model → Clients. The main server provides the calculated gradients to corresponding clients upon the completion of forward as well as backward passes, enabling them to update their local model parameters. The server also refines its own portion of the model and prepares the updated weights for synchronization with the Fed Server. By handling the more intensive training operations, the Main Server relieves client devices of intensive computation while maintaining distributed learning efficiency.

Fed Server: The fed server serves as the global coordinator of the entire SFL training process. It manages multiple main servers or directly multiple clients in smaller configurations, and aggregates their locally updated model parameters. The server aggregates such updates into a unified global model with the use of a federated averaging process, which is subsequently redistributed to every client. This preserves data privacy as well as communication effectiveness, also guaranteeing global model consistency throughout the distributed network. The Fed Server also oversees privacy control mechanisms, security authentication, and round synchronization among all clients.

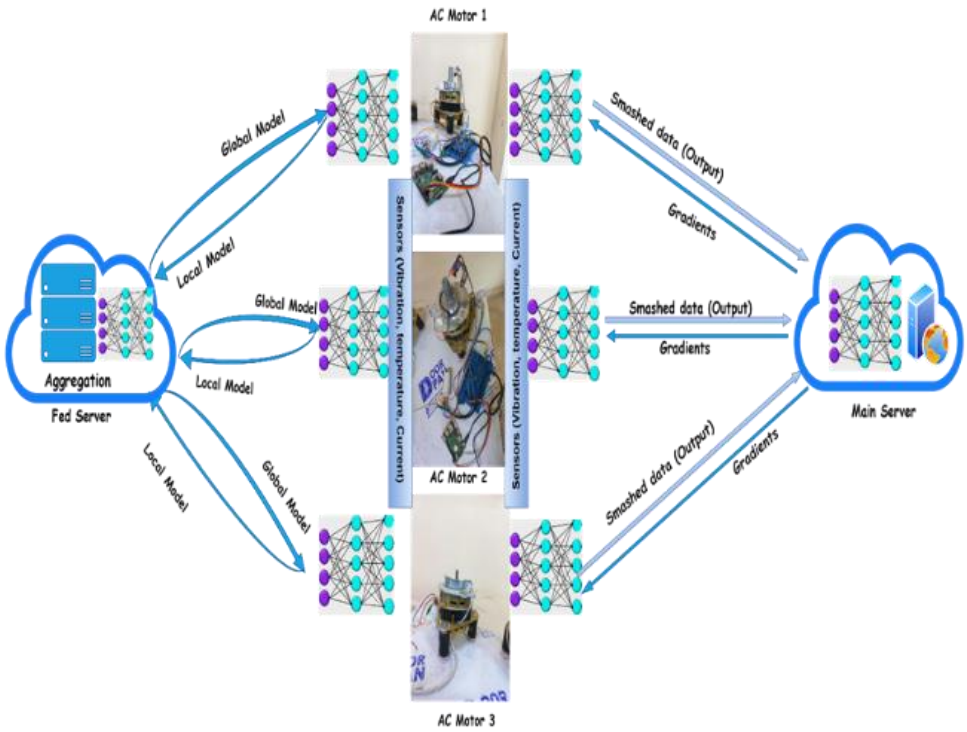


Figure 6. The proposed system architecture

6. RESULTS AND DISCUSSION

The presented section explains the experimental findings obtained from implementing three distinct learning frameworks: FL, SL, and the proposed SFL. The analysis compares their performance in terms of communication efficiency, prediction accuracy, and data privacy, with an emphasis on their applicability in industrial PdM environments.

In SL, model training is split between the central server and

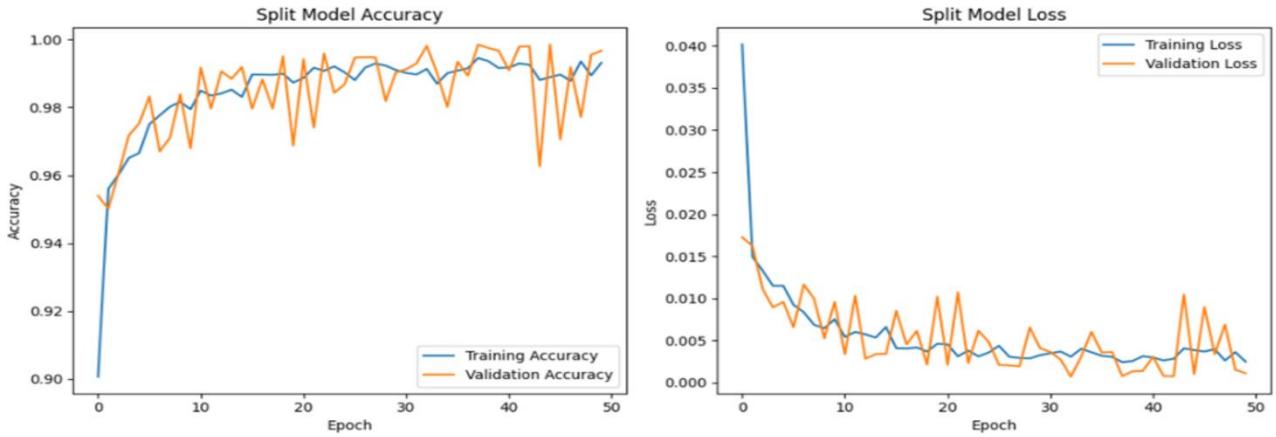
clients, allowing each one of the client devices to transmit only intermediate feature representations rather than raw data. This approach ensures a strong level of data privacy but introduces significant communication overhead, as frequent exchanges of activations and gradients occur during each training iteration. Consequently, SL demonstrates reliable accuracy but suffers from high latency; thus, it is less effective with regard to real-time or large-scale industrial applications. Figure 7 illustrates the training process related to the global model, which aggregates the contributions from three client models. In this setup, the cut layer was chosen as the last layer (output layer), where the intermediate activations from all clients are combined on the cloud server to finalize the forward pass.

Federated learning (FL), on the other hand, enables each one of the clients to train a complete local model as well as share only the updated parameters with the central server. Therefore, this reduces communication frequency and enhances scalability compared to SL. However, FL lacks feature-level collaboration, as the local training occurs in isolation, which can limit model generalization, especially when client data are highly heterogeneous. Figure 8 explains employing FL for training a local model on distributed data.

The proposed SFL framework integrates the advantages of both SL and FL. In this hybrid configuration, the DL model is partitioned so that clients handle initial feature extraction, while the main server performs deeper representation learning. The federated server then aggregates model parameters across all participating nodes, maintaining global consistency. This design significantly reduces communication overhead, improves convergence speed, and maintains strict data privacy, making SFL the most efficient and reliable approach for PdM applications in industrial environments. Table 4 shows the performance comparison between FL, SL, and the suggested SFL framework, where FL achieved a high accuracy, followed by SL, while SFL achieved slightly lower. However, SFL offers a balanced trade-off among computational load, communication efficiency, and privacy, making it more suitable for resource-constrained IIoT environments. The proposed SFL framework. This indicates that the model generalizes well to unseen data, making it adequate for PdM applications, in which reliable performance is of utmost importance, as shown in Figure 9.

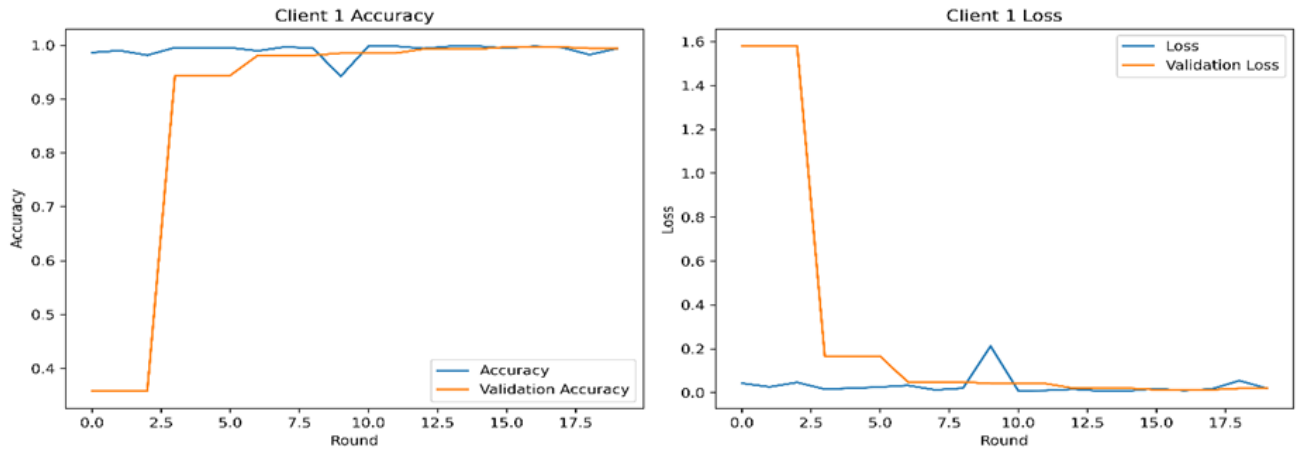
Table 4. Testing performance comparison between Split Learning (SL), Federated Learning (FL), and the proposed Split Federated Learning (SFL) framework

Model	Accuracy	Loss
SL	0.9935	0.0022
FL	0.9983	0.0104
SFL	0.9768	0.0804

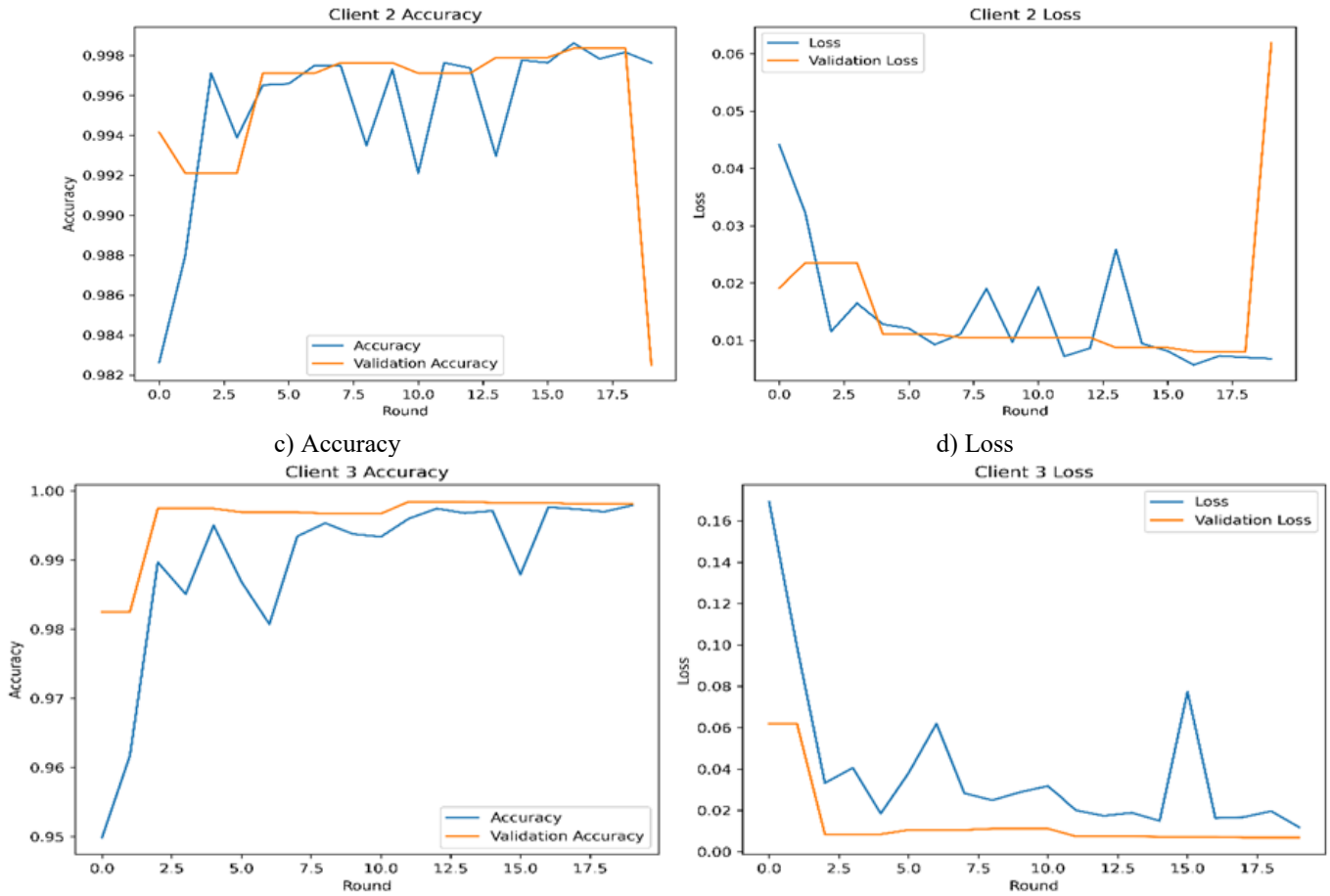


a) Accuracy b) Loss

Figure 7. Two types of performance metrics with a split (with Split Learning (SL)). a) Accuracy b) Loss



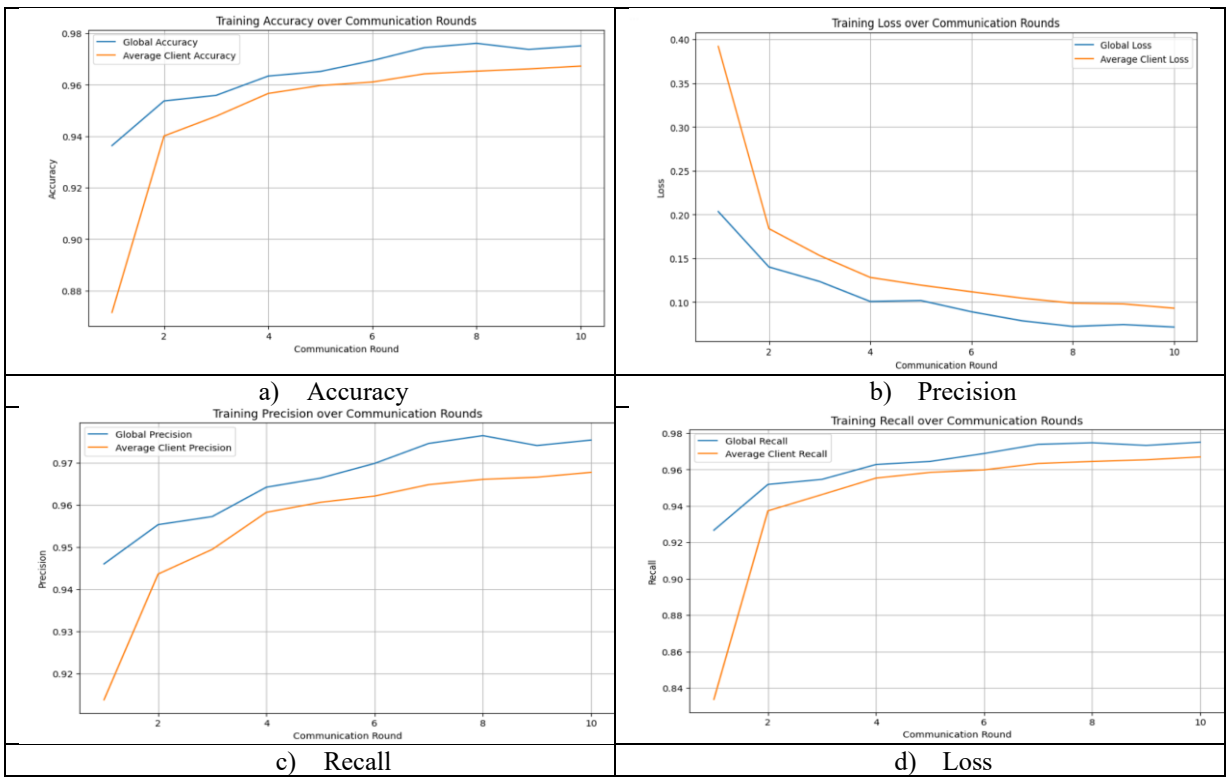
a) Accuracy b) Loss



c) Accuracy d) Loss

e) Accuracy f) Loss

Figure 8. DNN model performance for Stader system (with Federated Learning (FL))



a) Accuracy b) Precision

c) Recall d) Loss

Figure 9. Shows testing results of the Split Federated Learning (SFL) framework: a) Accuracy, b) Precision, c) Recall, d) Loss

7. CONCLUSION

In this study, we introduced SFL, a novel solution designed to tackle the challenges arising from heterogeneous data and resource landscapes in edge IoT environments. Empirical evaluation of SFL conducted on a real testbed of IoT devices has shown significant improvements in training time, accuracy, and energy usage, particularly when benchmarked against classic FL and SL approaches. These results underscore the efficacy of SFL in enhancing the performance of distributed learning IoT environments. By enabling more efficient and effective distributed model training, SFL has the potential to accelerate the adoption of FL in real-world IoT systems, paving the way for more intelligent, responsive, and efficient IoT solutions. These findings highlight that SFL offers a practical, efficient, and reliable solution for deploying PdM in real-world industrial environments. The SFL improves training efficiency and scalability, rather than absolute accuracy, and privacy, and provides robust predictive capabilities for unseen data. Future research will focus on enhancing the SFL framework for large-scale, heterogeneous industrial data, improving real-time fault detection, and integrating robust privacy and security measures for IIoT-based PdM applications.

REFERENCES

[1] Karuppusamy, D.P. (2020). Machine learning approach to predictive maintenance in the manufacturing industry - A comparative study. Journal of Soft Computing Paradigm, 2(4): 246-255. https://doi.org/10.36548/jscp.2020.4.006

[2] Zarzoor, A.R., Al-Jamali, N.A.S., Qader, D.A.A. (2023). Intrusion detection method for the Internet of Things based on the spiking neural network and decision tree method. International Journal of Electrical and Computer Engineering, 13(2): 2278. https://doi.org/10.11591/ijece.v13i2.pp2278-2288

[3] Al-Jamali, N.A.S., Al-Raweshidy, H.S. (2021). Smart IoT network-based convolutional recurrent neural network with element-wise prediction system. IEEE Access, 9: 47864-47874. https://doi.org/10.1109/ACCESS.2021.3068610

[4] Obaid, M.H., Hamad, A.H. (2024). Internet of things-based oil pipeline spill detection system using deep learning and LAB colour algorithm. Iraqi Journal for Electrical and Electronic Engineering, 20(1): 137-148. https://doi.org/10.18280/jesa.560416

[5] Mohammed, N.A., Abdulateef, O.F., Hamad, A.H. (2023). An IoT and machine learning-based predictive maintenance system for electrical motors. Journal Européen des Systèmes Automatisés, 56(4): 651-661. https://doi.org/10.18280/jesa.560414

[6] Tsai, Y.H., Chang, D.M., Hsu, T.C. (2022). Edge computing based on federated learning for machine monitoring. Applied Sciences, 12(10): 5178. https://doi.org/10.3390/app12105178

[7] Cakir, M., Guvenc, M.A., Mistikoglu, S. (2021). The experimental application of popular machine learning algorithms on predictive maintenance and the design of an IIoT-based condition monitoring system. Computers & Industrial Engineering, 151: 106948. https://doi.org/10.1016/j.cie.2020.106948

[8] Sohaib, M., Mushtaq, S., Uddin, J. (2021). Deep learning for data-driven predictive maintenance. In Vision, Sensing and Analytics: Integrative Approaches, pp. 71-95. http://doi.org/10.1007/978-3-030-75490-7_3

[9] Li, Z., He, Q., Li, J. (2024). A survey of deep learning-driven architecture for predictive maintenance. Engineering Applications of Artificial Intelligence, 133: 108285. https://doi.org/10.1016/j.engappai.2024.108285

[10] Duan, Q., Hu, S., Deng, R., Lu, Z. (2022). Combined federated and split learning in edge computing for ubiquitous intelligence in Internet of Things: State of the art and future directions. Sensors, 22(16): 5983. https://doi.org/10.3390/s22165983

[11] Thapa, C., Arachchige, P.C.M., Camtepe, S., Sun, L. (2022). Splitfed: When federated learning meets split learning. Proceedings of the AAAI Conference on Artificial Intelligence, 36(8): 8485-8493. https://doi.org/10.1609/aaai.v36i8.20825

[12] Zheng, G., Ivanov, D., Brintrup, A. (2025). An adaptive federated learning system for information sharing in supply chains. International Journal of Production Research, 63(11): 3938-3960. https://doi.org/10.1080/00207543.2024.2432469

[13] Jalali, A., Soltany, M., Greenspan, M., Etemad, A. (2025). Learning time-series representations by hierarchical uniformity-tolerance latent balancing. arXiv preprint arXiv:2510.01658. https://doi.org/10.48550/arXiv.2510.01658

[14] Hu, Z., Zhou, T., Wu, B., Chen, C., Wang, Y. (2025). A review and experimental evaluation of split learning. Future Internet, 17(2): 87. https://doi.org/10.3390/fi17020087

[15] Lokesh, G.H., Hukkeri, G.S., Jhanjhi, N.Z., Lin, H. (2024). Split Federated Learning for Secure IoT Applications: Concepts, frameworks, applications, and case studies. The Institution of Engineering and Technology. https://doi.org/10.1049/PBSE025E

[16] Li, A., Sun, J., Zeng, X., Zhang, M., Li, H., Chen, Y. (2021). Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In the 19th ACM Conference on Embedded Networked Sensor Systems, Coimbra Portugal, pp. 42-55. https://doi.org/10.1145/3485730.3485929

[17] Thapa, C., Chamikara, M.A.P., Camtepe, S.A. (2021). Advancements of federated learning towards privacy preservation: From federated learning to split learning. In Federated Learning Systems: Towards Next-Generation AI, pp. 79-109. https://doi.org/10.1007/978-3-030-70604-3_4

[18] Wang, C., Yang, Y., Zhou, P. (2020). Towards efficient scheduling of federated mobile devices under computational and statistical heterogeneity. IEEE Transactions on Parallel and Distributed Systems, 32(2): 394-410. https://doi.org/10.1109/TPDS.2020.3023905

[19] Turina, V., Zhang, Z., Esposito, F., Matta, I. (2021). Federated or split? A performance and privacy analysis of hybrid split and federated learning architectures. In 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), Chicago, IL, USA, pp. 250-260. https://doi.org/10.1109/CLOUD53861.2021.00038

[20] Joshi, P., Thapa, C., Hasanuzzaman, M., Scully, T., Afli, H. (2023). Federated split learning with only positive labels for a resource-constrained IoT environment. arXiv

preprint arXiv:2307.13266. https://doi.org/10.48550/arXiv.2307.13266
[21] Abedi, A., Khan, S.S. (2024). FedSL: Federated split learning on distributed sequential data in recurrent neural networks. Multimedia Tools and Applications, 83(10): 28891-28911. https://doi.org/10.1007/s11042-023-15184-5
[22] Hussien, H.A., Al-Messri, Z.A. (2023). Synthesis, characterization, and evaluation of some Meldrum's acid derivatives as lubricant additives. Iraqi Journal of Science, 1041-1048. https://doi.org/10.24996/ijs.2023.64.3.1
[23] Abood, R.H., Hamad, A.H. (2025). Multi-label diabetic retinopathy detection using transfer learning based convolutional neural network. Fusion: Practice and Applications, 279-293. https://doi.org/10.54216/FPA.170221
[24] Liu, M., Zeng, A., Chen, M., Xu, Z., Lai, Q., Ma, L., Xu, Q. (2022). Scinet: Time series modeling and forecasting with sample convolution and interaction. Advances in Neural Information Processing Systems, 35: 5816-5828. https://doi.org/10.52202/068431-0421